# KMC MODELING OF HELIUM BUBBLE CLUSTERING AND EVOLUTION IN BCC IRON

BY

AARON JAMESON OAKS

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Nuclear Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2015

Urbana, Illinois

Doctoral Committee:

Professor James F. Stubbins, Chair
Professor Brent Heuser
Professor Rizwan Uddin
Professor Robert Averback

# Abstract

The effect of helium in iron is an important issue in nuclear systems, as iron and iron alloys (steels) are the primary materials used for structural elements. Helium is known to cause embrittlement and decrease fatigue life, as well as aid creep and promote swelling. These effects can significantly alter the mechanical properties of the reactor materials, and generally lead to early failure and decreased part lifetimes. This is a concern in both fission and fusion systems. The precise role that helium, helium-vacancy clusters, and helium bubbles play in the material degradation processes described above are still only partially understood. Further understanding into the role helium plays in these phenomena is essential to predicting the lifetime of iron and steels in nuclear reactors.

This work was motivated by the results found earlier by Okuniewski. Said work was primarily experimental work studying the effects of helium concentration on cluster size distribution. KMC simulations were run for comparison, but the results were inconsistent. Both with and without helium present, the results showed the KMC simulation resulted in a significant shift compared to the experimental results. The KMC simulations predicted a high density of small sized clusters, while the experimental results showed a lower density of larger sized clusters. This inconsistency was believed to be a result of the various parameters chosen in the KMC model.

This work focused on two primary goals: first, to develop a flexible KMC code capable of simulating the desired models, and second, to explore the modeling assumptions made in the previous KMC simulations in an attempt to come closer to experimental results. Several different models for cluster interaction range, dissociation energy, and migration energy were considered, and a KMC code was designed and built to accommodate these and other models. The code design will be presented, along with performance benchmarking results. Both annealing and damage simulations were then performed with varying combinations of parameter models. The results of these simulations are compared and discussed.

*To my family, for their love and support.*

# Acknowledgments

I would like to give special thanks to my advisor, Prof. James F. Stubbins, for his continuous support on my work and his advice towards my professional development. I would also like to thank Prof. Heuser, Prof. Uddin, and Prof. Averback, for serving on my dissertation committee and providing useful feedback throughout the process.

I would like to thank my colleagues: Carolyn Tomchik, Di Yun, Bei Ye, and Wei-Ying Chen, for their support, contribution of ideas, and many helpful discussions relating to my work.

I would like to acknowledge the Illinois Campus Cluster Program for providing access to the computational resources used to perform the simulations in this study.

I would also like to acknowledge the funding sources for this work: NRC Fellowship, NRC Award NRC-38-08-967, and DOE NEUP Irradiation Performance of Fe-Cr Base Alloys DOE-INL Contract No. 00127139.

# Contents

# Chapter 1

# Introduction

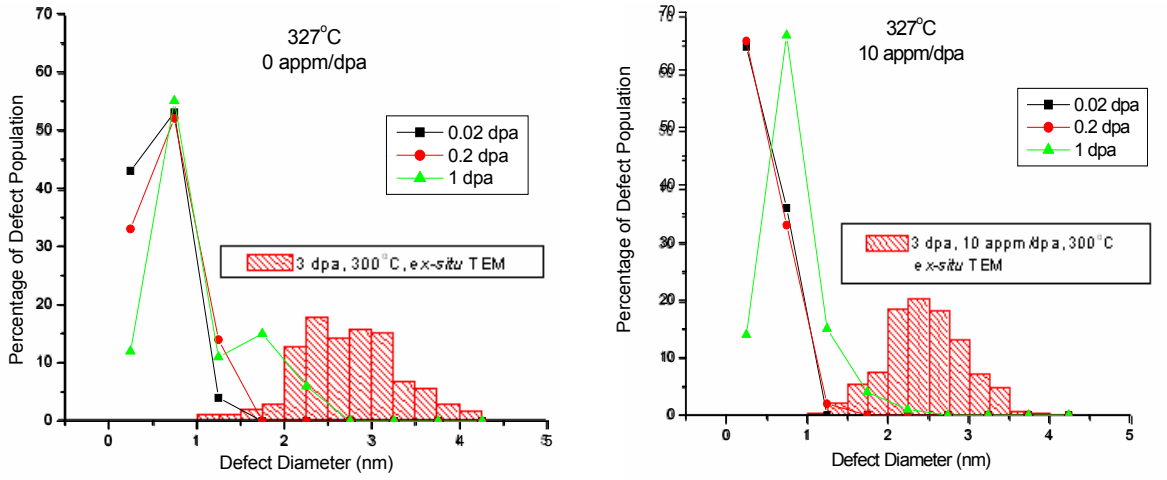
The effect of helium in iron is an important issue in nuclear systems, as iron and iron alloys (steels) are the primary materials used for structural elements. Helium is known to cause embrittlement and decrease fatigue life, as well as aid creep and promote swelling. These effects can significantly alter the mechanical properties of the reactor materials, and generally lead to early failure and decreased part lifetimes. This is a concern in both fission and fusion systems. In both cases the high neutron flux from the fission or fusion reactions causes $(n, \alpha)$ nuclear reactions in structural materials when interacting with the iron and alloy constituents [1]. This is of even greater concern in fusion reactors as helium is one of the primary products of the hydrogen fusion reactions, and unlike in fission reactors, the fuel (plasma) is not contained in any sort of cladding, allowing helium implantation directly into the first wall material [2]. Helium is highly insoluble in iron, and as a result tends to segregate and acts a precipitate [3]. In addition, helium can migrate to grain boundaries and precipitate into bubbles on these boundaries, causing high temperature embrittlement [4–6]. Void formation is also a very common occurrence in reactor materials. Helium can trap in these voids and stabilize them, further enhancing the aggregation of vacancies and helium atoms [7].

There have been numerous experimental studies aimed at studying the effects of helium in reactor steels. Grossbeck et al. performed a series of experiments looking at the fatigue life and creep rate in ferritic steels with He/dpa values characteristic of a fusion reactor first wall. Comparing irradiated and unirradiated specimens, they found that the irradiated specimens exhibited a reduction in fatigue life by a factor 3 to 10 compared to unirradiated material [8]. Comparing creep rates in materials with different helium concentrations, they found that specimens containing up to 200 appmHe experienced 3 to 10 times higher creep rate than similar specimens containing less than 20 appmHe [9]. Schroeder et al. carried out creep rupture tests on austenitic stainless steels with helium implanted at room temperature, at test temperature (~1073 K), and "in-beam" during the rupture test. They found that while the unirradiated control samples showed ductile behavior with transgranular failure, all of the He-implanted samples showed brittle, intergranular early failure [10]. They also found that embrittlement effect was enhanced for the "in-beam" tested samples. They attributed the difference between

the different treated sample sets to different helium bubble microstructures. The precise role that helium, helium-vacancy clusters, and helium bubbles play in the material degradation processes described above are still only partially understood. Further understanding into the role helium plays in these phenomena is essential to predicting the lifetime of iron and steels in nuclear reactors.

## 1.1 Motivation

This work was motivated by the results found earlier by Okuniewski [11]. Said work was primarily experimental work studying the effects of helium concentration on cluster size distribution. KMC simulations were run for comparison, but the results were inconsistent. These results are shown in fig. 1.1, both with and without helium present. In both cases, the KMC simulation results showed a significant shift compared to the experimental results. The KMC simulations (lines) predicted a high density of small sized clusters, while the experimental results (bars) showed a lower density of larger sized clusters.




**(a)** Without He **(b)** With He

**Figure 1.1:** Comparison of KMC simulation results (lines) with experiments (bars)[11]. Results show KMC model consistently underestimates cluster size distribution.

There are a number of models and assumptions that go into the formation of a KMC code, including clustering model, interaction distances, migration energies, dissociation energies, etc., and it was thought that one or more of the assumptions made in the modeling effort was the cause of the inconsistencies. Numerous parameter models are available, but the code used in the original study by Okuniewski was designed specifically for the models chosen, and was not

capable of model exploration. Thus, this work was designed with two primary goals. First, to develop a flexible KMC code capable of simulating the desired models, and second, to explore the modeling assumptions made in the previous KMC simulations in an attempt to come closer to experimental trends.

# Chapter 2

# Material Properties

## 2.1 Iron Systems

This work is aimed at studying the effects of helium in reactor steels, however, iron systems are quite complex. Figure 2.1 shows the phase diagram of the iron-carbon system. In only the 0-5 wt% carbon range, the phase varies widely with temperature, including numerous solid phases and mixtures of these phases. Most steels are more than iron-carbon alloys. Stainless steels provide improved corrosion resistance, and are primarily iron-chromium alloys, but typically contain several other alloy components for additional mechanical property improvements. For example, while 304 stainless steel is an Fe-Cr-Ni alloy, 306 stainless steel is a complex alloy of Fe-Cr-Ni-Mo-Mn-Si-S-N-C-P.

Due to the complexity of modeling the realistic steel systems, as well as the lack of reliable interaction potentials for the steel components, this study will focus on the pure iron system. At temperatures below 910 °C, $\alpha$-Fe ($\alpha$ ferrite) is the stable phase. $\alpha$ iron forms in the body centered cubic (BCC) structure, with a lattice parameter of $a_0 = 2.87$ Å. This is shown schematically in fig. 2.2. In this structure, each iron atom is coordinated by eight nearest-neighbor iron atoms at the cube corners (shown in fig. 2.2a), octahedral interstitials are located on the cube edges and cube face centers (shown in fig. 2.2b), and tetrahedral interstitials are located on the cube faces (shown in fig. 2.2c).

## 2.2 Defect Energetics

To begin the analysis of microstructure, numerous studies have been done to calculate the energetics of the three primary states of helium in iron: single helium atoms (interstitial), helium-vacancy clusters, and helium bubbles. One of the earliest studies by Rimmer and Cottrell studied the solubility of inert gas atoms in copper [13]. They concluded that two main configurations contributed to helium accumulation: interstitial helium because of its low

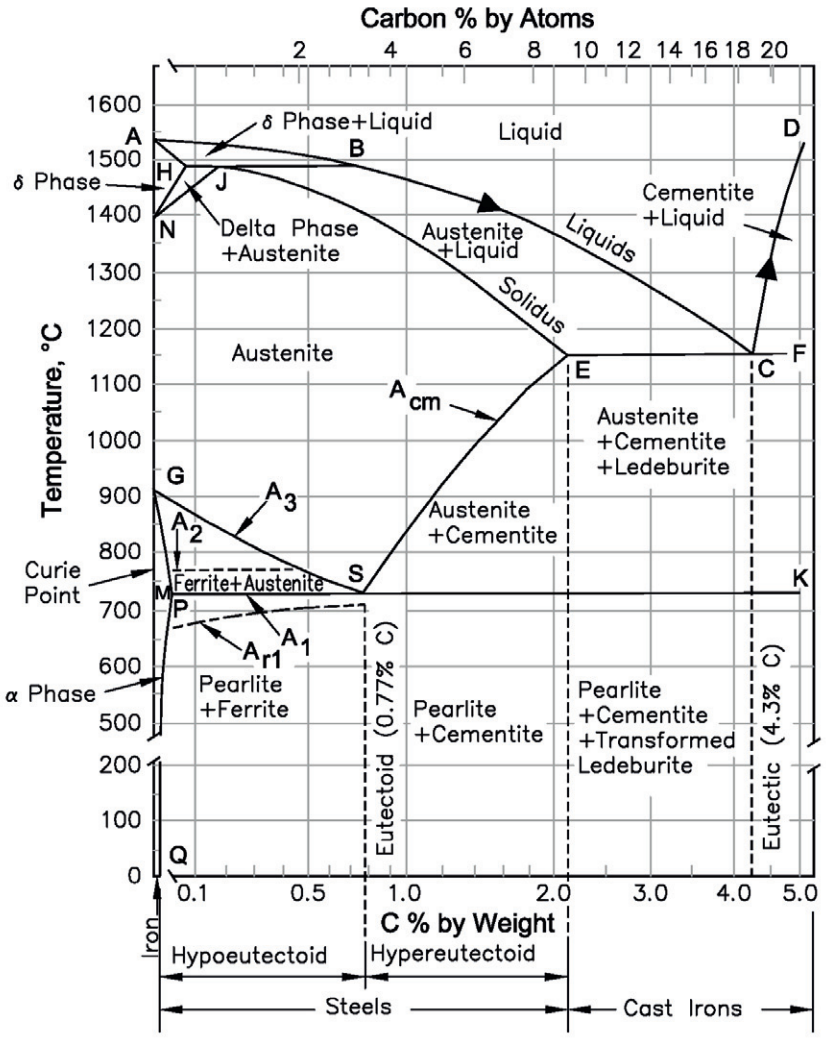
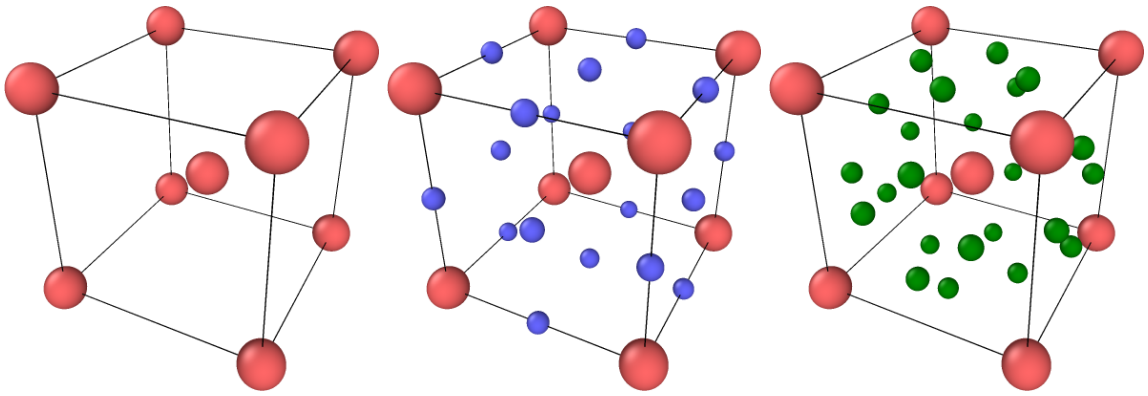



**Figure 2.1:** Iron-carbon phase diagram. [12]

migration energy, and substitutional helium because of its high trapping energy.

More recently, Density Functional Theory (DFT) calculations have been used to calculate helium dissolution and migration mechanisms in BCC iron. Fu et al. found that the tetrahedral configuration is energetically favorable compared to the octahedral configuration for interstitial helium atoms [14]. They also found that the energy difference between tetrahedral and substitutional He was unexpectedly small compared to previous empirical estimations. They attributed this discrepancy to a significantly lower binding energy between a vacancy and interstitial He. They also found that the migration of substitutional helium by the vacancy mechanism is governed by the migration of the $HeV_2$ cluster.

One major aspect that differentiates iron and iron-based systems from other reactor materials is its magnetism. Magnetism stabilizes the BCC crystal structure and affects the stability of



**(a)** Crystal lattice **(b)** Octahedral interstitial sites **(c)** Tetrahedral interstitial sites

**Figure 2.2:** Schematic representation of the body centered cubic (BCC) lattice. (a) BCC iron lattice. Iron atoms are located at the cube center and corners. (b) BCC octahedral sites. Iron atoms are indicated by red circles and octahedral interstitial positions are indicated by purple circles. Octahedral positions are are centered are each face and edge. (c) BCC tetrahedral sites. Iron atoms are indicated by red circles and tetrahedral interstitial positions are indicated by green circles. Tetrahedral positions form a diamond shape on each face.

any self-interstitial clusters formed. Accounting for the magnetic interactions complicates the modeling efforts, so investigations have been performed to determine the effect of magnetism on helium in an iron matrix. Seletskaia et al. performed DFT calculations on He in iron taking the magnetism into effect, and found that in contrast to previous calculations, the tetrahedral interstitial site was energetically more favorable than the octahedral interstitial site, which they attributed to the influence of a magnetism resulting from the defect's electronic structure [15]. This has been refuted by Fu et al. [14], who found that the changes in electronic structure of Fe from the insertion of He produced weak effects on the magnetization compared to other impurities or self-interstitials, and concluded that there was no evidence of a direct magnetic effect on the relative stabilities of He insertion sites. Zu et al. recently confirmed this conclusion, and found that the magnetism of the iron atoms does not directly affect the relative stability of the interstitial helium in BCC iron [16]. Thus, at zero Kelvin, the He-Fe potential does not explicitly need to include magnetic effects.

While DFT generally provides the most accurate energetics data, its computational complexity makes it suited only for relatively small systems (on the order of 100 atoms). To investigate larger system sizes and simulation times, Molecular Dynamics (MD) simulations are used. Unlike DFT, which calculates the electronic structure of an atomic system directly from quantum theory, MD evolves an atomic system according to classical Newtonian mechanics, according to given interatomic potential energy functions. These interatomic potentials generally follow specific forms, with fitting parameters that are adjusted to each type of atomic interaction.

Interactions generally need to be specified for every type of elemental interaction type, which are Fe-Fe, Fe-He, He-He in this system. These interactions are often pairwise, however more complicated multibody potential forms exist which take additional interaction effects into account.

A review by Samaras [17] provides a comparison of calculated energies for a collection of interaction potentials, including the Wilson [18], Seletskaia [19], and Juslin-Nordlund (J-N) [20] Fe-He potentials, the Ackland [21], Dudarev-Derlet (D-D) [22], and Mendelev [23] Fe-Fe potentials, and the Beck He-He potential [24]. A comparison of defect formation energies for various combinations of potentials is given in table 2.1. It is clear from the table that certain potentials more accurately reproduce the DFT results than others. Earlier Fe-He potentials like Wilson and Seletskaia predict the octahedral site as the more stable interstitial position, instead of the tetrahedral site predicted by DFT. Newer Fe-He potentials have been fitted in such a way that the tetrahedral site is the energetically favorable site, in agreement with DFT calculations. It is also clear that even with a fixed Fe-He potential, the results vary based on available Fe-Fe potentials. The effects of magnetism affects the formation of interstitial defects [14, 25, 26]. In these results, using the Juslin-Nordlund Fe-He potential, the Mendelev potential best reproduced the defect formation energies, followed by Dudarev-Derlet and Ackland.

**Table 2.1:** Formation energies of substitutional and interstitial helium atoms in an iron matrix. The first two rows are DFT calculations for comparison [14, 15]. The remaining rows are MD calculations using a variety of Fe-Fe, Fe-He interaction potentials. All simulations were performed with the Beck He-He potential [24]. AMS refers to Mendelev [23], J-N refers to Juslin-Nordlund [20], D-D refers to Dudarev-Derlet [22], and FS refers to Ackland [21] (Finnis Sinclair type). Energies are in eV.

| | Fe-He | Fe-Fe | Oct | Tetr | Subs |
|---|---|---|---|---|---|
| **DFT** | Seletskaia | | 4.60 | 4.37 | 4.08 |
| | Fu | | 4.57 | 4.39 | 4.22 |
| **MD** | J-N | AMS | 4.512 | 4.385 | 4.099 |
| | J-N | D-D | 4.444 | 4.326 | 4.212 |
| | J-N | FS | 4.406 | 4.29 | 4.116 |
| | Wilson | FS | 5.25 | 5.34 | 3.25 |
| | Seletskaia | FS | 4.54 | 4.5 | 3.91 |

## 2.3 Helium-Vacancy Clusters

The formation of small helium-vacancy clusters has been studied in numerous molecular dynamics simulations using different interaction potentials. When using Ackland-Wilson-Beck

(Fe-Fe, Fe-He, He-He interactions) potentials, both Morishita et al. [27] and Ventelon et al. [28] found that interstitial helium jumps into iron vacancy sites, becoming substitutional and forming a He-V cluster with a large binding energy, which agrees with DFT calculations [28]. Morishita also found that because the helium is so strongly bound to the He-V cluster, it can stabilize the cluster and increase its lifetime by reducing thermal vacancy emission. As the He/V ratio of the cluster grows, the vacancy dissociation energy grows from ~1 eV at an He/V ratio of 0 to ~7 eV at an He/V ratio of 6 [27]. This trend is confirmed by the DFT calculations from Fu et al. [14], who also showed that small He-V clusters (containing up to 4 helium atoms) will lead to bubble nucleation in initial vacancy-free lattices.

Using their own Fe-He potential, Wilson et al. found that five helium atoms in the iron matrix induces the kick-out of an iron atom, forming a near-Frenkel pair [18]. This kick out phenomenon also shows the importance of choosing the correct potential for simulations. When using the Wilson Fe-He potential, the iron knock-out occurs when the He interstitials are at far distances, whereas when using Juslin-Nordlund (J-N) Fe-He potential, the iron knock-out occurs when the He interstitials are in close proximity [20]. Yang et al. also compared the Wilson and Seletskaia Fe-He potentials to show the difference in point defect creation and cluster creation [29]. They found that since the Seletskaia potential overestimates the iron interstitial formation energy, the kick-out of iron interstitials will occur less frequently than with the Wilson potential. Wirth et al. also found that in the $HeV_2$ di-vacancy cluster, the helium atom prefers to sit in a symmetric location between the two vacancies, rather than at the substitutional site of either vacancy [30].

## 2.4 Helium Bubbles

Since helium gas pressure can stabilize voids, there have been numerous studies on the energetics of helium bubbles. Using the Ackland-Wilson-Beck (A-W-B) potential, Morishita et al. found that the binding energy of the He-V clusters was independent of the cluster size, but was instead dependent on the helium-to-vacancy (He/V) ratio [27]. They found that the binding energy increased with more helium, causing the cluster lifetime to increase by reducing thermal vacancy emission. In a different study [31], they also found that when the He/V ratio is high ($He/V > 10$), the bubble pressure becomes sufficiently large that there is spontaneous creating of vacancies due to their low formation energy. They found that this leads to thermal emission of both SIA iron atoms, SIA clusters, and helium atoms.

As with point defects, when comparing results from MD with DFT calculations there are discrepancies based on which interaction potential combinations are used. When studying small

He-V clusters with DFT, Fu et al. calculated that the most stable clusters had a 1.3 He/V ratio with a He dissociation energy of 2.6 eV [32]. Morishita et al. performed similar studies using MD and the Ackland-Wilson-Beck (A-W-B) potential combination [27]. Their resulted predicted a 1.6 He/V ratio with a He dissociation energy of 3.6 eV. Lucas et al. carried out a comparison study of dissociation energies surrounding He-V clusters with A-W-B, A-JN-B, DD-W-B, DD-JN-B potentials [33]. They found that only the A-JN-B (Ackland-Juslin-Nordlund-Beck) potential combination agreed with DFT calculations.

Numerous studies have also been done looking at the energetics of larger clusters. As might be expected from ideal behavior, both Trinkaus et al. [34] and Walsh et al. [35] confirmed that as the He-V bubble size increases, the equilibrium helium pressure decreases. It has also been shown consistently through MD simulations that larger helium clusters have lower He/V ratios, and emit SIA iron atoms more easily [33, 36, 37]. Haghighat et al. also studies the shape of larger He/V clusters (A-W-B potential combination) and found that the stable bubbles take on surface polyhedron morphology [38].

## 2.5 Radiation Damage

Reactor steels are exposed to high fluxes of neutron irradiation over a range of temperatures, which as discussed earlier produces both radiation damage defects and helium as a results of $(n, \alpha)$ reactions. It is therefore important to understand how the irradiation produces defects and how the radiation affects defect structures. Radiation effects simulations are typically done by simulating a collision cascade using molecular dynamics. Yang et al. carried out a series of cascade studies (A-W-B potential combination), looking at iron with pre-existing substitutional helium ranging from 1-5%, with He-V clusters, with temperatures ranging from 100-600 K, and with PKA energies ranging from 500 eV-40 keV [39–42]. They found that helium does not affect the Frenkel pair production and that there was no difference in interstitial or substitutional helium during the collision state. Morishita et al. found that since substitutional helium atoms move to neighboring vacancies very easily, helium will migrate quickly during a damage melt when many Frenkel pairs are produced [27]. They also found that helium atoms will migrate quickly if they are knocked out of substitutional positions into interstitial positions in the collision cascade. If the fast moving helium atoms are able to move to the edge of the cascade region then they will not recombine with the vacancies at the region center. Schaublin et al. also found that He atoms can be trapped by iron interstitials, preventing them for combining with vacancies to form substitutional He, resulting in vacancies remaining in the iron matrix after collision [43].

Studies were also carried out to look at the effects of irradiation on pre-existing He-V clusters. The results tend to show that the resulting damage depends on the He/V ratio in the cluster (based on A-W-B potential simulations). Pu et al. found that a high He/V ratio leads to a larger vacancy cluster since Frenkel pair recombination is not possible [44]. Yang et al. found that small He/V ratio leads to helium dissolution and recombination of vacancies with neighboring iron atoms [41]. These results are consistent with DFT calculations done by Fu et al., who found that clusters with large He/V ratios tend to emit He interstitials, while clusters with small He/V ratios tend to emit vacancies [32].

## 2.6 Potentials

There were numerous potentials used throughout the literature for Fe-Fe, Fe-He, and He-He interactions, the most popular being the (A-W-B) combination. However, based on the potential review study by Lucas [33], only the combination of the Ackland Fe-Fe potential [21], the Juslin-Nordlund (J-N) Fe-He potential [20], and the Beck He-He potential reproduced helium bubble energy results consistent with DFT calculations, so these models are reviewed here.

All of the simulation works found used the Beck He-He potential [24] given by eq. (2.1):

$$V_{He-He}(r_{ij}) = A\exp\left(-\alpha r_{ij} - \beta r_{ij}^6\right) - \frac{0.869}{\left(r_{ij}^2 + a^2\right)^3}\left(1 + \frac{2.709 + 3a^2}{r_{ij}^2 + a^2}\right) \tag{2.1}$$

The pair potential parameters are given in table 2.2.

**Table 2.2:** Pair potential parameters for the Beck He-He interaction potential.

| Parameter | Value |
|---|---|
| a (Å) | 0.675 |
| $\alpha$ (Å$^{-1}$) | 4.39 |
| $\beta$ (Å$^{-6}$) | $3.746 \times 10^{-4}$ |
| A (eV) | 398.7 |

The J-N Fe-He potential [20] is a piecewise pair potential given by eq. (2.2):

$$V_{Fe-He}\left(r_{ij}\right) = \begin{cases} \text{DMOL-potential} & r_{ij} \le r_1 \\ p_3 r_{ij}^3 + p_2 r_{ij}^2 + p_1 r_{ij} + p_0 & r_1 \le r_{ij} \le r_2 \\ \left(a + \frac{b}{r_{ij}}\right) e^{-c r_{ij}} f_c\left(r_{ij}\right) & r_{ij} \ge r_2 \end{cases} \tag{2.2}$$

$$f_c\left(r_{ij}\right) = \begin{cases} 1 & r_{ij} \leq r_c - r_d \\ \frac{1}{2}\left(1 - \sin\frac{\pi\left(r_{ij}-r_c\right)}{2r_d}\right) & \left|r_c - r_{ij}\right| \leq r_d \\ 0 & r_{ij} \geq r_c + r_d \end{cases} \tag{2.3}$$

where $f_c\left(r_{ij}\right)$ is a cutoff function included for computation efficiency given in eq. (2.3), the pair potential parameters are given in table 2.3, and DMOL-potential is a tabular form DMOL dimer potential evaluated from 0.001 Å to 1.000 Å in steps of 0.001 Å, with values given in the potential reference. These values must be interpolated (typically using cubic splines) as part of the potential initialization. Juslin notes that the DMOL dimer term can be substituted by the ZBL universal screened-Coulomb potential [45], as both are similar. The choice does not affect the potential part fitted for larger distances. The polynomial term in the middle range is introduced to ensure smooth values and derivatives at the transition points between the two potential forms.

**Table 2.3:** Pair potential parameters for the Juslin-Nordlund Fe-He interaction potential.

| Parameter | Value | Parameter | Value | Parameter | Value |
|---|---|---|---|---|---|
| a (eV) | 26.65 | $r_1$ (Å) | 1.0 | $p_3$ (eV Å$^{-3}$) | 62.020897 |
| b (eV Å) | -15.0 | $r_2$ (Å) | 1.2 | $p_2$ (eV Å$^{-2}$) | -96.287579 |
| c (Å$^{-1}$) | 1.856 | $r_c$ (Å) | 3.7 | $p_1$ (eV Å$^{-1}$) | -38.548739 |
| | | $r_d$ (Å) | 0.25 | $p_0$ (eV) | 79.266283 |

The Ackland Fe-Fe potential is a combination of the ZBL universal screened-Coulomb potential [45] for very short range interactions and a many-body potential for larger range interactions [21]. The many-body potential is presented in the framework of the Finnis-Sinclair, where the energy of an assembly of $N$ atoms is given by eq. (2.4):

$$E = \frac{1}{2}\sum_{i\neq j=1}^{N} V(x_{ij}) - \sum_{i=1}^{N}\left(\sum_{j=1}^{N}\Phi\left(x_{ij}\right)\right)^{1/2} \tag{2.4}$$

where $V_{ij}$ is the pairwise repulsive contribution of the potential, and $\Phi_{ij}$ in the many-body, isotropic, cohesive term is also a pairwise function. These functions were fitted to (computationally-convenient) cubic splines, given in eqs. (2.5) and (2.6):

$$V(r) = \sum_{k=1}^{m} a_k H(r_k - r)(r_k - r)^3 \tag{2.5}$$

$$\Phi(r) = \sum_{k=1}^{2} A_k H(R_k - r)(R_k - r)^3 \tag{2.6}$$

where in this notation, $H(x)$ is the Heaviside step function. The fitting parameters are given in table 2.4.

**Table 2.4:** Potential parameters for the Ackland many-body potential.

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| $a_1$ (eV/$a_0^3$) | -36.559853 | $r_1$ ($a_0$) | 1.180000 |
| $a_2$ (eV/$a_0^3$) | 62.416005 | $r_2$ ($a_0$) | 1.150000 |
| $a_3$ (eV/$a_0^3$) | -13.155649 | $r_3$ ($a_0$) | 1.080000 |
| $a_4$ (eV/$a_0^3$) | -2.721376 | $r_4$ ($a_0$) | 0.990000 |
| $a_5$ (eV/$a_0^3$) | 8.7619863 | $r_5$ ($a_0$) | 0.930000 |
| $a_6$ (eV/$a_0^3$) | 100.0000 | $r_6$ ($a_0$) | 0.866025 |
| $A_1$ (eV$^2$/$a_0^3$) | 72.868366 | $R_1$ ($a_0$) | 1.300000 |
| $A_2$ (eV$^2$/$a_0^3$) | -100.944815 | $R_2$ ($a_0$) | 1.200000 |
| $a_0$ (Å) | 2.8665 | | |

In order to study atomic collisions in radiation damage, the potential must be adjusted for distances inside the normal nearest-neighbor spacing. This is done by replacing the many-body potential with the ZBL universal screened-Coulomb potential [45] for short ranges, and adding in a transition function to smooth the potential as it transitions between the two potential forms, given in eq. (2.7).

$$V_{Fe-Fe}(r_{ij}) = \begin{cases} \frac{Z_{Fe} Z_{Fe} e^2}{r_{ij}} \phi\left(\frac{r_{ij}}{a_s}\right) & r_{ij} \leq r_1 \\ \exp\left(B_0 + B_1 r_{ij} + B_2 r_{ij}^2 + B_3 r_{ij}^3\right) & r_1 \leq r_{ij} \leq r_2 \\ \sum_{j=1}^{N} V(r_{ij}) - \left(\sum_{j=1}^{N} \Phi\left(r_{ij}\right)\right)^{1/2} & r_{ij} \geq r_2 \end{cases} \tag{2.7}$$

where the ZBL universal screening function $\phi(x)$ and screening length $a_s$ are given by eq. (2.8) and eq. (2.9), respectively:

$$\phi(x) = 0.1818e^{-3.2x} + 0.5099e^{-0.9423x} + 0.2802e^{-0.4029x} + 0.02817e^{-0.2016x} \tag{2.8}$$

$$a_s = \frac{0.88534 a_b}{\left(Z_{Fe}^{2/3} + Z_{Fe}^{2/3}\right)^{1/2}} \quad , \quad a_b = 0.529\,\text{Å} \tag{2.9}$$

where $a_b$ is the Bohr radius. The fitting parameters for the transition function and transition points are given in table 2.5.

**Table 2.5:** Transition function fitting parameters for ZBL-modified Ackland potential.

| **Parameter** | **Value** |
|---|---|
| $B_0$ | 7.14705133 |
| $B_1$ ($\text{Å}^{-1}$) | 0.69010282 |
| $B_2$ ($\text{Å}^{-2}$) | -4.16604662 |
| $B_3$ ($\text{Å}^{-3}$) | 1.06871772 |
| $x_1$ (Å) | 0.90 |
| $x_2$ (Å) | 1.90 |

# Chapter 3

# KMC Code Development

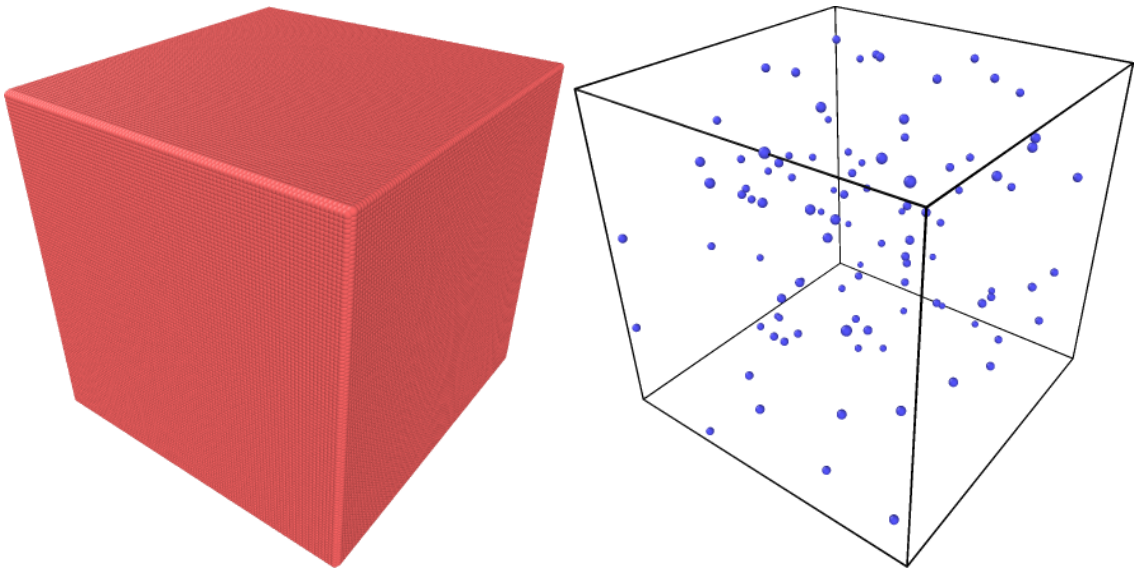
## 3.1 Motivation

Ideally, atomic simulation would be done with Molecular Dynamics (MD), which evolves the system according to Newton's laws and is based directly on the interatomic potentials of the interacting species ($\vec{F} = \nabla U$). Assuming the interatomic potentials are correct, this would give the most accurate evolution of the atoms in the system. Unfortunately, due to their integrative nature, MD simulations suffer from time scale limitations, usually limited to evolving on the order of picoseconds, and possibly up to nanoseconds. As a result, it is generally difficult or impossible to use MD to study longer term diffusion or defect structure evolution, or to compare results with experimental results. Kinetic Monte Carlo (KMC) provides a way to solve this problem. By simulating events such as atomic jumps as unit events, the time scale can be extended by orders of magnitude compared to the small atomic vibrations simulated in MD.

Another major benefit to using KMC is that since the atomic migrations are parameterized and act as unit events, the simulation can be done on the defects in the system, rather than the entire system of atoms. As an example, consider a simulation to study the behavior of 100 vacancies in a 2 million atom system. When simulated using MD, the only things that are simulated are atoms. As a result, to simulate the example system, the code would need to simulate the 1 999 900 remaining atoms. In KMC, since migrations are assumed to be unit events, simulating the migration of an atom from an occupied position to a vacant position is equivalent to simulating the migration of a vacancy from the vacant position to the occupied position. As a result, the simulation can assume a background lattice of 2 million atoms, and only simulate the 100 vacancies. This is shown visually in fig. 3.1.

Kinetic Monte Carlo simulations follow a relatively straightforward algorithm. For a system where some $N$ processes (in this case primarily atomic migrations) can occur with known rates $r_i$, the KMC evolution of the system is governed by the following algorithm:

1. Initialize simulation time to $t = 0$.



**(a)** Example MD lattice. **(b)** Example KMC lattice.

**Figure 3.1:** Illustration of full MD simulations compared with defect KMC simulations for 2 million lattice site system with 100 vacancies. The MD system (a) simulates 1 999 900 iron atoms (red). The equivalent KMC system (b) simulates only 100 vacancies (purple).

2. Form a list of all possible rates in the system $r_i$.
3. Calculate the cumulative function $R_i = \sum_{j=1}^{i} r_i$ for each rate $i = 1, \ldots, N$. Let $R = R_N$ be the total rate sum. (This can be thought of as calculating a discrete cumulative distribution function from the discrete probability density function $r_i$, although this is not technically true because the probability density function is not normalized.)
4. Generate a uniform random number $\rho_1 \in (0, 1]$.
5. Find the event $i$ to carry out by finding the $i$ such that $R_{i-1} < \rho_1 R \leq R_i$. (This can be thought of as inverting the previously calculated cumulative distribution function.)
6. Carry out event $i$.
7. Generate a new random number $\rho_2 \in (0, 1]$.
8. Update the simulation time with $t = t + \Delta t$, where $\Delta t = -\ln(\rho_2)/R$.
9. Recalculate all rates $r_i$ that may have changed due the event that was carried out. If necessary, remove or add any new rates to the list of possible rates.
10. Return to step 3 and repeat until the desired end condition is reached.

Unfortunately, despite the simplicity of the algorithm, the implementation of certain steps (specifically items 2 and 6) are quite complex, and generally depend heavily on the system and models being simulated. As a result, a major aspect of this work was the development of a KMC code that could properly simulate the systems and models of interest.

## 3.2 KMC Code

In order to simulate the desired iron systems, as well as explore the various parameters that influence the evolution of the system, a custom-built KMC code was required. To facilitate this, a KMC code that was created previously by the author was used as a foundation for further development [46]. This code was originally developed to study the influence on dopant concentrations on the mobility of defects in the system, specifically how the presence of lanthanum affects the mobility of oxygen in cerium oxide. As a result, the code focused primarily on single particle migrations with migration energies that varied based on specific nearest neighbor cation configurations. While that code could not simulate any of the clustering behavior necessary for this study, it was designed to be flexible and extensible, and was thus chosen for the basis of development for this study. This section will cover the general design of the base code, as well as the work done to extend it to support the clustering and parameter models used in this study.

### 3.2.1 General Design

The KMC code developed in this study is written in object-oriented C++, and makes a best effort to minimize unnecessary external dependences. The goal was to allow the code to be built and deployed without significant concern for the build environment on the target system, as configurations tend to vary widely between sites. The bulk of the data structures are implemented using standard STL containers (e.g., vectors, lists, maps, etc.) and language built-ins. Several notable exceptions include the use of C++11 (C++0x) for random number generation, the libxml2 library for XML data transport, and the OpenMP library for parallelization. While these are external dependencies, they are each relatively standard. C++11 and OpenMP functionality are available in most modern C++ compilers, and the libxml2 library is commonly found on most computational systems. The design decisions to include these dependencies are explained in sections 3.2.4, 3.2.5 and 3.5.2.

### 3.2.2 Program Flow

The basic flow of the code is described here. In these descriptions certain words are capitalized to indicate that they refer to the corresponding data object in the code: Entity, Action, Event, Cluster.

The code starts by reading in the particle information from an input file. This file defines all of the entities that can exist in the system, and provides details such as particle sublattice, Frenkel pair relationships, and cluster relationships. By having this information read in as input, the code is able to adapt to different simulation systems and defects without any internal modification or rebuilding. An example entity input file is given in listing 3.1. As the code processes this file and creates internal entity identifiers, it creates bi-directional maps between the names and identifiers. This serves to create context for the remaining data I/O, as the maps allow for logical particle names to be used in the input and output files instead of their internal numerical representation (e.g. the interstitial helium particle type can be represented by the logical name `int_helium` instead of its internal numerical identifier 4).

```
<?xml version="1.0" encoding="utf-8"?>
<root>
  <entities>
    <entity>
      <name>I1</name>
      <sublattice>substitutional</sublattice>
      <capture_radius>7.06126653505</capture_radius>
      <frenkel_pair>He0V1</frenkel_pair>
    </entity>
    <entity>
      <name>He0V1</name>
      <sublattice>substitutional</sublattice>
      <capture_radius>6.14023176961</capture_radius>
      <frenkel_pair>I1</frenkel_pair>
    </entity>
    <entity>
      <name>He1V0</name>
      <sublattice>octahedral</sublattice>
      <capture_radius>6.07536586492</capture_radius>
    </entity>
  </entities>
</root>
```

**Listing 3.1:** Example entity information input file. Cluster relationships have been removed for clarity.

After the entity information has been read in, the code reads in the simulation configuration from a separate input file. These include values like simulation system dimensions, system temperature, end conditions, and particle populations. An example configuration file is given in listing 3.2. Having this information specified separately from the entity information allows

the code to be easily run on a series of input configurations that are based on the same entity information without having to duplicate that information. Once the configuration file has been parsed, the simulation system is set up and initialized.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <boltzmann_constant>8.61738e-05</boltzmann_constant>
  <lattice_parameter>2.87e-08</lattice_parameter>
  <lattice_type>body_centered_cubic</lattice_type>
  <simulation_dpa>0.03</simulation_dpa>
  <end_condition>dpa</end_condition>
  <temperature>573</temperature>
  <dimensions>
    <x>125</x>
    <y>125</y>
    <z>125</z>
  </dimensions>
  <initial_populations>
    <population>
      <entity>I1</entity>
      <count>703</count>
    </population>
    <population>
      <entity>He1V0</entity>
      <count>72</count>
    </population>
    <population>
      <entity>He0V1</entity>
      <count>703</count>
    </population>
  </initial_populations>
</configuration>
```

**Listing 3.2:** Example simulation configuration information input file.

Next, the action information is read in from a third input file. These include values like action directions, attempt frequencies, and energies. Again, having this information specified in a separate file allows the allowable events and energies to vary without having to repeat entity or configuration information. An example action file is given in listing 3.3. Originally, this information was used to create a list of Action objects, but this method proved inefficient for the current simulation. As a result, additional bookkeeping structures are also created to improve efficiency. This is discussed in detail in section 3.5.1.

To complete initialization of the system, the initial population information from the configuration file is used to populate the system. For each entity type that has an initial population, Entity objects are created for the specified number of that object, distributed randomly throughout the simulation system.

```
<?xml version="1.0" encoding="utf-8"?>
<actions>
  <action>
    <entity>He0V1</entity>
    <sublattice>substitutional</sublattice>
    <type>migration</type>
    <frequency>6e+12</frequency>
    <energy>0.69</energy>
    <direction>
      <x>2</x>
      <y>-2</y>
      <z>2</z>
    </direction>
  </action>
  <action>
    <entity>He1V0</entity>
    <sublattice>octahedral</sublattice>
    <type>migration</type>
    <frequency>6e+12</frequency>
    <energy>0.078</energy>
    <direction>
      <x>0</x>
      <y>-2</y>
      <z>0</z>
    </direction>
  </action>
</actions>
```

**Listing 3.3:** Example action information input file.

After both the Entity list and Action list have been initialized, the system is ready to be evolved. Originally, an event catalog of all of the possible migration events is generated by scanning through the Entity list and Action list and checking each possible combination. This was modified to incorporate the bookkeeping data structures mentioned earlier, which will be discussed further in section 3.5.1. For each combination of Entity and Action, a list of checks is performed to see if the pair is a valid migration event. This includes checking that the particle type that the action acts upon and the particle type of the paired Entity match and checking that the sublattice that the actions acts upon and the sublattice of the paired particle match. This is necessary as the lists can contain Entities and Actions for different particles on different sublattices, and it is important that the Entity and Actions selected for an Event are compatible. A check is also performed on the proposed final position of the particle to ensure that the final position is currently unoccupied, as moving a particle to a position where a particle already exists is not possible in this model. If an Entity/Action pair passes all of the checks, it is considered a possible event, and an Event object consisting of the Entity/Action pair is added to the event catalog.

Once the list of all possible events has been generated, a single event is chosen to be carried out. The partial sums of the entity rates are calculated and an event is randomly selected. The event is carried out by applying the Action to the Entity's position and updating the Entity object's internal position information. Once the event has finished, the Entity is checked against other Entities in the system for possible interactions based on the cluster model, which will be discussed further in section 3.3. Finally, the simulation time and DPA level is then updated. At this point, the event catalog is no longer consistent with the system, so it is emptied in preparation for the next time step. The system loops over this process until the desired end condition is reached. Once the system has finished its evolution, run statistics are generated and output for analysis. The algorithm for this process is shown visually in fig. 3.2.

### 3.2.3 Simulation End Conditions

The code supports several end conditions, based on the type of simulation being run. These determine when to end the simulation and output final simulation statistics, as well as determine the points for time series output, which is discussed further in section 3.2.6. The end condition for a given simulation is specified in the configuration file, and so can be varied without changing the code. Perhaps the simplest and most intuitive end condition is KMC steps. Using this condition, the simulation will always end after a specific number of KMC steps have been performed. This is very useful during development, and when testing the efficiency of different parts of the algorithm (see section 3.5). For more realistic comparisons of simulations the code also supports time and DPA levels as end conditions. Since simulation time is tracked as part of the KMC algorithm, it is fairly straightforward to adapt the end control and time series output to time-based control. DPA level, on the other hand, is more complicated. DPA level needs to be calculated throughout the simulation, although since only certain events represent damage, this simulation can be made more efficient by only updating DPA calculation only after such events. The time series output takes more effort than the other conditions. Unlike KMC step and simulation time, which are unique for each step, the DPA level does not necessary change with each step. This would cause problems if the time series output system used a simple approach to checking if output should be performed (i.e., $timeSeriesStep = endTime/outputPoints$), as all steps after the critical damage event would match this condition until the next damage event occurred. To solve this problem, the output conditions are calculated before starting the simulation, and an additional bookkeeping structure is added to track whether output has occurred for specific DPA levels.

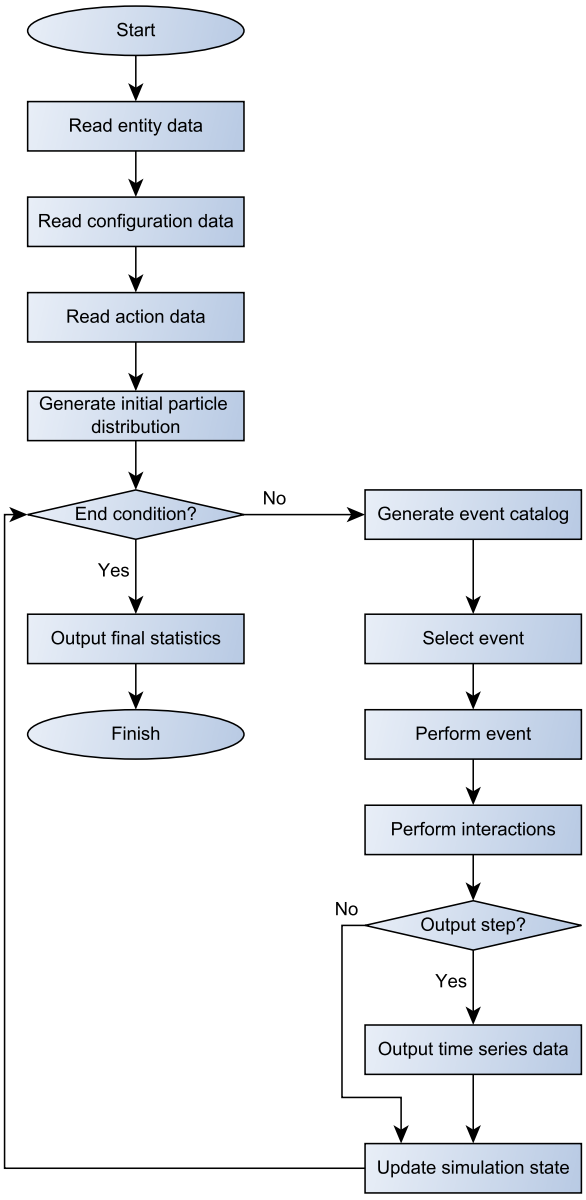



**Figure 3.2:** Overall program algorithm flowchart.

### 3.2.4 Random Number Generation

As a stochastic simulation, it is important to discuss the source of random numbers in the simulation. Random numbers are used in a number of places throughout the simulations, so it is important that they be of high quality. Previous iterations of the code (and many C++ codes in general) are based on the C standard library `rand()` function, which is not a very good source of random numbers in this context [47]. In many implementations, it has a very limited range of $[0, 32767]$, meaning that random numbers will have a granularity of $1/32767 \approx 3 \times 10^{-5}$. This

is problematic in a KMC code where event rates may vary by that many orders of magnitude, as this could result in low probability events never being selected. It is also based on a linear congruential generator function, which has a number of drawbacks. Numbers generated by LC generators can also fall on hyperplanes depending on the parameters used in the generator. This can be a major problem for position generation, as particles that are supposed to be distributed randomly in the system may end up placed on specific planes in the 3-dimensional space. LC generators also typically have a relatively short period (typically $2^{32} \approx 4.3 \times 10^9$), and will start repeating themselves in large simulations. This can also be problematic in this code, as depending on how frequently particles are added or removed, the simulation may start repeating sequences of events.

These problems were solved by the introduction of more advanced random number generation in the C++11 standard. This standard introduced the Mersenne Twister 19937 generator as part of the C++ standard, which was previously only available as part of large add-on libraries such as Boost. The addition of this generator to the standard means that this code can be deployed without a dependence on such large external libraries, instead only depending on C++11 compiler support, which has become widely available (more so than the presence of add-on libraries on deployment targets). The MT generator solves the issues mentioned previously, as it has a range of $[0, 2^{32})$, which gives a granularity of $1/2^{32} \approx 2.3 \times 10^{-10}$, has a period of $2^{19937} - 1 \approx 4 \times 10^{6001}$ (will basically never repeat), and passes numerous tests for statistical randomness (e.g., no hyperplanes).

The C++11 standard also introduced the idea of distribution functions to C++. Again, historically using a random number generator to sample a distribution was left up to the developer, which often resulted in biased implementations (e.g., uniform distributions that are skewed towards lower numbers). This was a problem primarily with generating integers (e.g., for generating positions on an integer lattice), but also with generating real numbers (e.g., for selected events to perform), as biased distributions would bias the selection of locations, events, etc. C++11 provides template functions for a variety of uniform, normal, and Poisson distributions, among others. The uniform integer and uniform real distributions, along with the MT19937 random number generator, are used extensively throughout this code.

Finally, for the sake of reproducibility, the random number generator is seeded using the system time in milliseconds past the Unix epoch, which is also now available from the C++11 standard. The seed is chosen with a fine granularity to avoid starting multiple simulations with the same seed. The RNG used in the simulation is output with the other simulation output, and can be read in from the configuration file if specified, allowing runs to be reproduced exactly if necessary. This is especially useful during development, but can also be useful for verification, should the results need to be reproduced.

### 3.2.5 XML Data Transport

The code performs data I/O primarily in XML format. There are a number of benefits to this design decision. XML information is self-descriptive, so the structure and information can be easily parsed by readers without using a specifically written parser. XML schemas can be written for arbitrary data transport, which makes it ideal for customized data transport in codes such as this. The XML protocol is also very well supported, with XML interfaces either built into or readily available for almost every commonly used programming language. This makes using the code significantly easier, as one does not need to go through the effort of developing generators and parsers to read or write input files or output files. Such processes are tedious and error-prone, even for the developer.

Having all of the data transport performed in XML means that all of the work for reading and writing information can be handled by the XML implementation of the language in use. For example, to generate the data and relationships of clusters in a simulation, a script can be written to programmatically calculate the required data and create input files in XML format. An example of this process written in Python is shown in listing 3.4, which can be used to generate entity information similar to the example in listing 3.1. Similarly, once the code has run and generated output data, said data can be easily parsed and accessed. An example of this process written in R is shown in listing 3.5. The only tedious XML parsing occurs in the C++ code, in order to maintain portability. There are a number of XML parsers written for the C++, but the most standard implementation is libxml2. Unlike the other implementations, libxml2 is a C standard, is actively maintained, and should be available on nearly every deployment target. The only real drawback is that it is a C library (not C++), and as such does not take advantage of any of the convenient language constructs from C++. This makes reading and writing XML in the code more tedious, but the other benefits of libxml2 outweigh this drawback.

### 3.2.6 Time Series Output

The code supports time series output at regular intervals, currently corresponding to percent completion. The two primary time series of interest are particle/cluster position data and population distributions. While the population distribution should be derivable from the position data, the data processing and analysis are considerably easier when the populations are output together. In addition, outputting this information separate from the main simulation output avoids wasting precious memory. If the time series data was to be output with the main simulation output file, the data from each time series output step would need to be stored in memory until the simulation ends. When saving the population distribution and positions of

```
import xml.etree.ElementTree as ET
rootNode = ET.Element('root')
clustersNode = ET.SubElement(rootNode, "clusters")
...
# define He-V clusters
for numHe in range(0, maxHe+1):
    for numV in range(1, maxV+1):
        name = "He" + str(numHe) + "V" + str(numV)
        radius = calcRadius(V=numV, He=numHe)
        clusterNode = ET.SubElement(clustersNode, "cluster")

        addNode(clusterNode, "name", name) # entity name
        addNode(clusterNode, "sublattice", "substitutional")
        addTextNode(clusterNode, "capture_radius", str(radius)) # calculated radius

        populationsNode = ET.SubElement(clusterNode, "populations")
        populationNode = ET.SubElement(populationsNode, "population")
        addTextNode(populationNode, "entity", "He1V0")
        addTextNode(populationNode, "count", str(numHe)) # number of He in cluster
        populationNode = ET.SubElement(populationsNode, "population")
        addTextNode(populationNode, "entity", "He0V1")
        addTextNode(populationNode, "count", str(numV)) # number of V in cluster
tree = ET.ElementTree(rootNode)
tree.write("entities.xml") # write XML to file
```

**Listing 3.4:** Example Python script to generate input files based on desired parameter sets. The XML tree is represented as an object and can be easily build using the XML library functions.

```
library('XML')
xmlData = xmlParse("output.xml")
listData = xmlToList(xmlData)
runtimeMS = listData[["run_statistics"]][["run_time_ms"]] # get sim runtime in ms
...
```

**Listing 3.5:** Example R script to read and analyze output data files. The XML tree can be converted to a native list data structure, which can be easily accessed by name using the language accessors.

every particle in the system, this would consume a considerable amount of memory, and limit the bounds of the simulation. By outputting this data to separate files, the data can be written to disk as soon as the time series output step occurs, eliminating the need to store copies of the data in memory.

While most data I/O is done using XML, time series data is more conveniently handled using different formats. The population distributions are output in comma separated value (CSV) format with a header line that describes the column information. Like with XML, CSV files are easily handled by most programming languages, and can be quickly read in and converted to a language-specific table-like object. The first several columns identify simulation details

at the time of the population output, specifically KMC step, simulation time, and DPA level, if appropriate. After the identification data, the populations for all entity types are specified in the order they were specified in the entities input file. This can result in columns of zeros for entities that were defined but never used, but these can easily be filtered out during post processing. An example populations file is given for reference in listing 3.6.

```
timestep,time,He0V1,He1V0,He0V2,He0V3,He0V4,He0V5,He0V6,He0V7,He0V8
0,4.62E-16,3613,147,488,94,16,0,0,0,0
20000,1.52E-08,3272,0,528,110,23,0,0,0,0
40000,1.59E-07,2355,0,709,228,45,13,3,0,0
60000,3.34E-07,1859,0,699,312,80,28,9,0,0
80000,5.43E-07,1434,0,694,385,112,34,10,2,1
100000,7.92E-07,1184,0,621,424,150,50,13,3,2
120000,1.09E-06,953,0,555,460,176,65,19,5,3
140000,1.44E-06,790,0,489,461,199,87,29,7,2
160000,1.86E-06,617,0,407,470,220,107,38,11,4
180000,2.39E-06,486,0,337,475,233,125,47,16,4
200000,3.06E-06,370,0,266,454,246,150,58,22,5
220000,3.93E-06,273,0,210,438,247,162,75,30,7
240000,5.11E-06,206,0,145,421,250,172,87,34,13
260000,6.79E-06,134,0,99,395,257,176,99,40,17
280000,9.40E-06,89,0,62,385,254,169,110,50,22
300000,1.39E-05,42,0,25,365,243,176,118,49,30
320000,2.68E-05,15,0,3,340,226,190,112,57,35
340000,6.61E-05,7,0,1,286,200,192,125,64,43
360000,0.000104318,1,0,4,231,179,189,125,73,47
```

**Listing 3.6:** Example populations CSV file.

Particle position data is output using the LAMMPS DUMP format [48]. This format is similar to CSV, in that most of the data is position information and is stored in a delimited format (except using spaces instead of commas). However, this format also include header information for each timestep, including time, number of atoms, and simulation system size. This format was specifically chosen because it works very well for visualization, as a majority of atomistic simulation visualization software supports the DUMP format. LAMMPS utilities like Pizza.py [49] can also be used to parse and manipulate DUMP data, as well as transform it into other formats if necessary. An example DUMP file is given for reference in listing 3.7.

## 3.3 Clustering Model

The original code had focused on single particle migrations. These migrations were influenced by their surroundings so there was some interaction between them, but they did not combine

```
ITEM: TIMESTEP
4.62467e-16
ITEM: NUMBER OF ATOMS
4410
ITEM: BOX BOUNDS
0 125
0 125
0 125
ITEM: ATOMS element x y z
He0V1 3 15 31
He0V1 63 5 38
He0V1 29.5 26.5 30.5
He0V1 21 7 100
He0V1 106 46 48
He0V1 104 68 44
He0V1 103 111 7
He0V1 21 77 105
He0V1 24 70 15
He0V1 69 92 57
He0V1 57 112 33
```

**Listing 3.7:** Example positions DUMP file.

into clusters and act as aggregate entities. As a result, the primary focus of the code extension was to implement a clustering model that would allow void and bubble behavior to be studied.

### 3.3.1 Previous Work

Previous work has been done using both modeling and experimental techniques to investigate the vacancy cluster size distribution. Okuniewski carried out an extensive experimental study of cluster evolution in iron both with and without helium [11]. This experimental study was complimented by a KMC model developed by Deo et al. [50]. The results of this computational effort were compared with the experimental results, and Okuniewski found that the Deo KMC model consistently underestimated the cluster sizes and overestimated the small cluster densities compared to the experimental results. The experimental work found a vacancy cluster distribution consisting of a low density of large clusters with very few small clusters, while the Deo KMC model found a vacancy cluster distribution consisting of a very high density of small clusters with very few large clusters. Okuniewski concluded that improvements were needed in the KMC model in forms that might shift the cluster distribution away from a large number of small clusters towards a small number of large clusters, in order to bring the model into agreement with experimental results.

An improved cluster interaction model was proposed previously by Ortiz et al. [51]. While

the Deo model assumes a constant cluster interaction distance, the Ortiz model allows the interaction distance to vary as a function of cluster size. Ortiz performed isochronal annealing simulations and compared the results of the KMC model with a rate theory model, finding differences at low temperatures, but near perfect agreement at higher temperatures. This good agreement with other simulation techniques motivated a comparison between the Deo and Ortiz cluster interaction models to determine if the Ortiz model would predict a vacancy cluster size distribution in better agreement with the experimental results from Okuniewski. In order to facilitate this comparison, along with the other comparisons performed in this study, the original code needed to be significantly extended to support a clustering model.

### 3.3.2 Model Details

As is commonly accepted in the literature [52–54], defects in this model are represented as point defects with an interaction volume meant to represent the strain field of the defect, approximated by a sphere. Two defects are considered to be interacting when their interaction volumes overlap (see fig. 3.3).

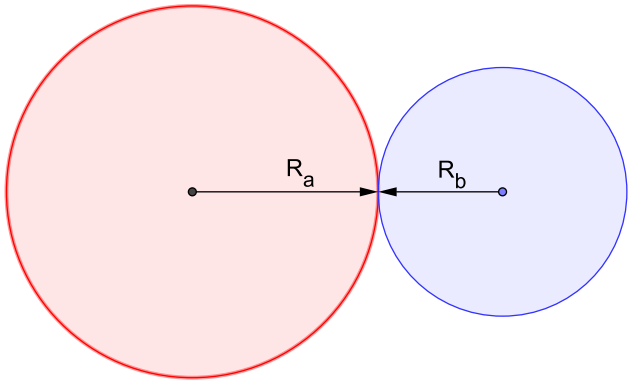



**Figure 3.3:** Defect Clustering Schematic. Defect interaction volume is approximated by a sphere of given radius. Two particles are considered to be interacting when their interaction volumes overlap.

The model supports both Frenkel pair recombination and defect clustering. While overlapping interaction volumes are considering "interacting", not all interactions result in a meaningful combination (i.e., some combinations of entities should not form clusters). To solve this, the Frenkel pair relationships, as well as the specifications for all meaningful clusters, are given as input to the model. When two particles are considered to be "interacting", the Frenkel pair recombination and cluster specifications are checked to see if this interaction is meaningful. If the interaction is meaningful, the interaction is handled accordingly (annihilation in the case of Frenkel pair recombination, or combination in the case of cluster formation), and

if the interaction is not meaningful, it is ignored. The model supports cluster formation (particle+particle) cluster growth (particle+cluster), and cluster coalescence (cluster+cluster). In each of these cases, Frenkel pair recombination is also considered and handled accordingly. This allows for the following cluster interaction types. Note that this is a list of interaction mechanisms, not an exhaustive list of all interactions. For example, in the case of cluster formation, only $V_2$ cluster formation is listed, but the mechanism supports the formation of all valid initial clusters (e.g., $I_2$, $He_2$, HeV, etc.):

$$
\begin{aligned}
I + V &\rightarrow 0 && \text{FP Recombination} \\
V + V &\rightarrow V_2 && \text{Formation} \\
V_n + V &\rightarrow V_{n+1} && \text{Growth} \\
V_n + I &\rightarrow V_{n-1} && \text{Shrinkage} \\
V_n + V_m &\rightarrow V_{n+m} && \text{Coalescence} \\
V_n + I_m &\rightarrow V_{n-m} && \text{Coalescence} \\
V_n + He_m &\rightarrow V_n He_m && \text{Coalescence}
\end{aligned}
$$

There are several important consequences of modeling clusters in this manner. Of primary concern is that since the clusters are represented as point particles, the internal structure of the cluster is not simulated. As a consequence, events that involve intracluster migration or rearrangement are not permitted. Since the goal of this study is not to study the internal structure of the clusters being formed, this is not a major drawback, however, it does impact the way that cluster mobility is handled. Vacancy clusters typically migrate through a surface diffusion mechanism [52], where rather than all vacancies in the cluster moving by some amount, the vacancies (or equivalently, lattice atoms) on one side of cluster migrate along the surface of the cluster to the other side, producing net diffusion. This is shown schematically in fig. 3.4. While indeed this mechanism cannot be simulated directly in the code, the net effect of the mechanism can parameterized and simulated. If it is known (or assumed), that a cluster will diffuse in this manner, a lower level simulation (in MD, for example) can be performed to study a single "migration event". The results of this simulation can be used to determine an effective migration energy for the cluster, which can then be used in the KMC simulation. This allows clusters be represented as point particles, but keep the underlying physics of the surface diffusion mechanism intact.

A second matter of concern is the simulation of larger SIA defects. While small SIA clusters can be considered spherical, larger SIA clusters are usually thought to be 2D loops. This clustering model assumes that all defect clusters, including large SIA clusters, act with a 3D

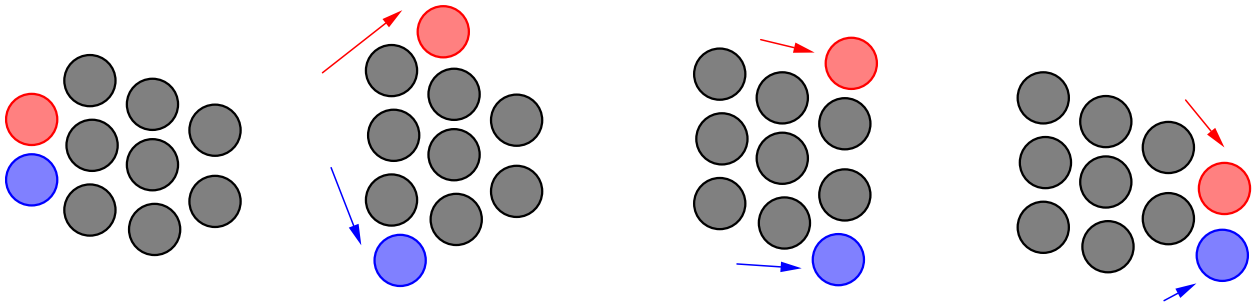


**Figure 3.4:** Schematic representation of cluster surface diffusion mechanism. Particles move along the surface from one side to another, resulting in net diffusion of the cluster. This is shown schematically by the red and blue particles moving around the stationary gray particles.

interaction volume. This is commonly done in other clustering models [52, 53, 55], as while a loop itself is a 2D object, the strain field it creates is 3D and roughly round (see fig. 3.5), so the assumption of a spherical interaction volume is not unreasonable.

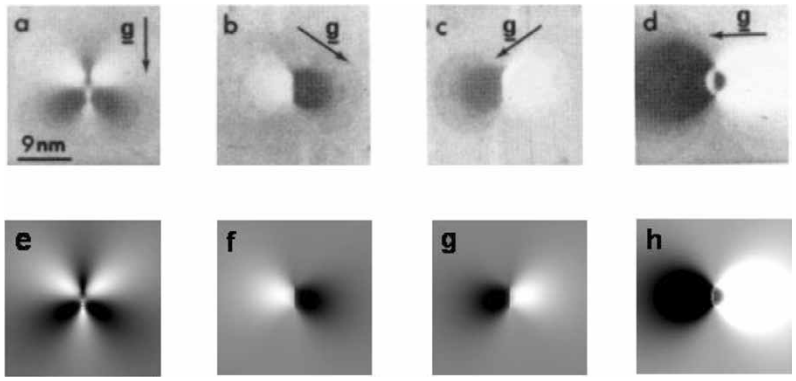



**Figure 3.5:** Simulated dark-field images of a perfect dislocation loop with $\vec{b} = 1/2[0\,1\,\bar{1}]$ under two-beam dynamical conditions. Images (a-d) are from Eyre et al.[56] and images (e-h) are from Zhou et al. [57]. (a) and (e) $\vec{g} = 2\,0\,0$, $\vec{g} \cdot \vec{b} = 0$; (b) and (f) $\vec{g} = 1\,1\,1$, $\vec{g} \cdot \vec{b} = 1$; (c) and (g) $\vec{g} = 1\,\bar{1}\,1$, $\vec{g} \cdot \vec{b} = 1$; (d) and (h) $\vec{g} = 0\,\bar{2}\,2$, $\vec{g} \cdot \vec{b} = 2$. Loop diameter is ~5 nm.

### 3.3.3 Model Implementation

Clusters are implemented in the code as a subclass of the Entity class, as Clusters interact with the rest of the code in basically the same manner, but have the added property that they contain multiple single Entity objects. As an Entity-type object, Clusters have and ID associated with them, but unlike Entity IDs, the Cluster IDs correspond to a listing of the cluster's population. As mentioned earlier, not every combination of defects forms an acceptable cluster. The acceptable clusters are specified as a set of population maps indicating how many of each defect type are

included. For example, an SIA and He atom might not form a cluster, but a set of two vacancies and an He atom might form the $HeV_2$ cluster, which would have a population map of {He:1, V:2}. This is shown schematically in fig. 3.6. The inclusion of a map corresponding to the list population may seem redundant, as the information should be available from scanning the list of Entities in the cluster, and for existing clusters, this is true. However, this sort of scan would not be possible during the formation of a cluster, as no such list would exist at that point. The population map is the only way to verify that the proposed cluster should actually form, and also speeds up comparing and computing populations for existing clusters during the interacts process.

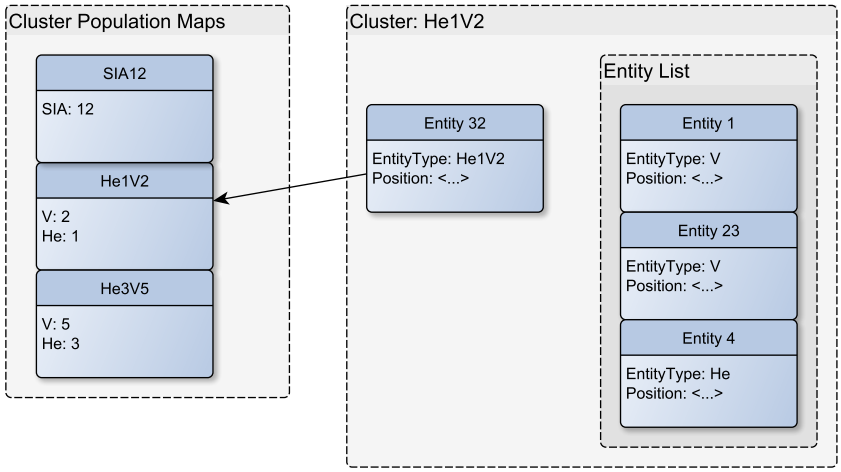



**Figure 3.6:** Schematic representation of Cluster object. Each Cluster contains standard Entity data on itself, a list of Entity objects that it represents, as well as a link to a corresponding population map.

The clustering model is implemented in the code by adding in an extra step at the end of the standard KMC algorithm loop to resolve any cluster interactions in the new environment created as a result of the KMC event that has just occurred. As defect interactions are essentially pairwise (each defect can theoretically interact with any other defect), a naive algorithm would do a pairwise scan of the entire system, checking every particle against every other particle, which would result in quadratic efficiency. However, since the system only moves one particle for each KMC step, this can be optimized. When the system is initialized, each defect that is added to the system is checked for interactions against all particles that have previously been added. This is effectively the quadratic method mentioned, but is only performed once when the system is being initialized. Once this is complete, the system is fully resolved. After this point, since only one particle can move for each KMC event, the only interactions that can change are between the event particle and the rest of the system. Thus, the scan for clusters need only consider interactions involving the event particle, which is linear complexity.

When processing interactions, the Entity that has just interacted is compared pairwise with all other Entities in the system (both single particles and clusters). For each pair, the pair is first checked to see if they two Entities are within interaction range. If not, the pair is ignored, and the next pair is checked, but if the pair is within range, the interaction process continues. There is a fairly complex logic tree that is used to properly handle the various types of interactions, as well as to manage the Cluster objects that exist in the system. At the top level, the pairwise interacting objects fall into one of three categories: particle-particle, particle-cluster, and cluster-cluster. These interactions determine the work that needs to be done to properly form or combine clusters, as well as clean up clusters that have been removed. These processes are described below.

### 3.3.4 Particle-Particle Interactions

When two single particles are within interaction range, there are three possible outcomes: they form a Frenkel pair and annihilate, they merge together to form a cluster, or they do not interact. To allow for Frenkel pair recombination, each defect type stores it's corresponding Frenkel pair type, if one exists. An example of this is shown in the example entity input file in listing 3.1. When two particles interact, they are first checked to see if they are a Frenkel pair. If they are, they recombine and are removed from the system. These particles still remain in memory so that they can be part of final simulation calculations (e.g., average displacement, diffusion, etc.), but are no longer able to participate in KMC events, or interact with other Entities in the system.

If the particles are not Frenkel pairs, the interaction continues to cluster formation. To determine if the two particles form a cluster, a proposed cluster map is created, and then populated with the values from the two particles. For example, if vacancy were in range of a He atom, the proposed cluster would map would be {He:1, V:1}. This proposed cluster is then checked against the list of acceptable clusters which were defined in the entity input file. If the proposed cluster is not found then the two particles do not form an acceptable cluster, and the interaction process is complete for that pair. If the proposed cluster is acceptable, a new Cluster object is created, assigned the appropriate ID, populated with the two particles, and added to the list of Entities in the system considered for KMC events. The particles that are added to the cluster have their positions set to the Cluster position (as clusters are assumed by the model to be point defects), and if necessary, the positions of Cluster and constituent entities are shifted to an appropriate sublattice site. In addition, since the Cluster that is created has its own potential for interaction, the interaction procedure is repeated recursively on the newly

created Cluster before proceeding. The algorithm for this process is shown visually in fig. 3.7.

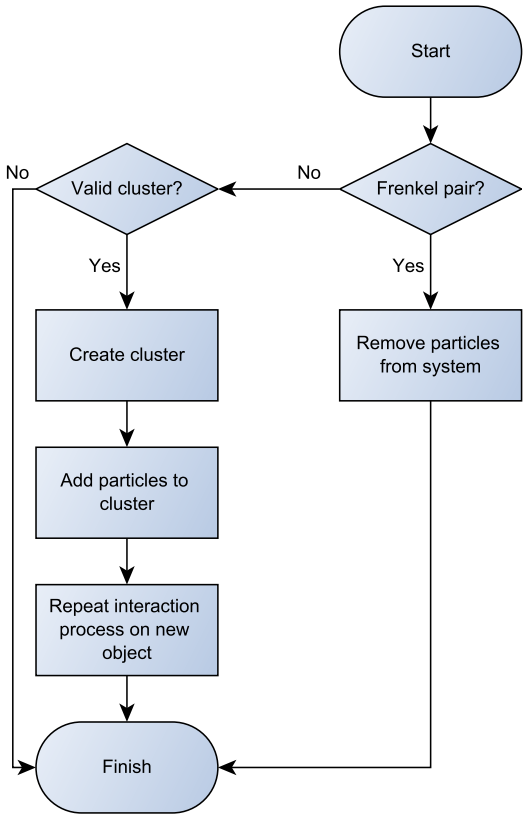



**Figure 3.7:** Particle-cluster interactions algorithm flowchart.

### 3.3.5 Particle-Cluster Interactions

When a single particle and cluster interact, the procedure for resolution flows similarly to that of particle-particle interactions, but in reverse order. As with particle-particle interactions a proposed cluster population map is created by retrieving the map for the interacting cluster, then adding in the interacting particle. The map is first checked for Frenkel pairs, and reduced if any are found. For example, if an SIA were to interact with an $HeV_3$, the map would start as {He:1, V:3, I:1}, then be reduced to {He:1, V:2}. Once the population has been reduced, it is checked against the list of acceptable clusters. This must be done after reduction, as the list only stores fully reduced representations of clusters, which would cause the lookup to fail. For example, $HeV_3I$ may be considered equivalent to $HeV_2$, but the acceptable cluster list would only contain $HeV_2$, and would fail to find a $HeV_3I$ cluster, indicating that the clustering interaction should not proceed, when in fact it should. Like with particle-particle interactions,

if the acceptable cluster lookup fails, the interaction is considered invalid and skipped. If the proposed cluster is acceptable, the process continues.

The single particle is added to the Cluster, its position is set to the cluster position. If there were Frenkel pairs reduced during the validation stage, the interacting Frenkel pairs are now properly annihilated as with particle-particle Frenkel pairs. If this process results in a Cluster that still contains more than one constituent Entity, the cluster ID is updated to its new value. If not, then the interaction produced what is essentially a single particle cluster (e.g., $I+V_2 \longrightarrow V$), which is not allowed. In this case, the now single particle is removed from the cluster, and the now empty cluster is deleted from the system. Again, since whatever this process produces (either Cluster or single Entity) has its own potential for interaction, the interaction procedure is repeated recursively on the newly object before proceeding. The algorithm for this process is shown visually in fig. 3.8.

### 3.3.6 Cluster-Cluster Interactions

When two clusters interact, the procedure is very similar to the particle-cluster interaction, but with two specific variations. First, when a particle interacts with a cluster, the particle is always moved to the cluster's position. However, when a cluster interacts with another cluster, it is not obvious which should "join" the other. In this algorithm, the number of particles in each cluster is calculated, and the smaller cluster is merged into the larger cluster. Second, when a particle interacts with a cluster, it was possible (through Frenkel pair recombination) to finish with a single entity cluster. This is still possible in cluster-cluster interactions, but it is now also possible for clusters to completely annihilate (e.g., $V_2 + I_2 \longrightarrow 0$). This possibility is also checked, and if it occurs, the resulting zero-size cluster is deleted from the system. The algorithm for this process is shown visually in fig. 3.9.

## 3.4 Supported Event Types

The original code was developed to study single particle migration, so this was the only event type available. In order to properly simulate the cluster model used in this study, additional event types were implemented.

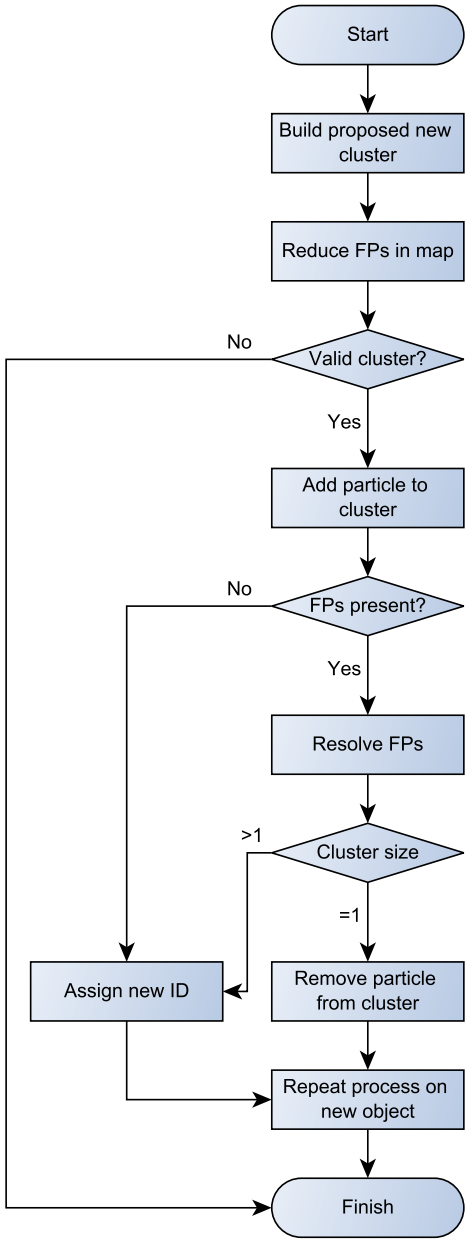



**Figure 3.8:** Particle-cluster interactions algorithm flowchart.

### 3.4.1 Particle Migration

Single particle migration is the basic event available, with most other events based upon it. There are a number of checks that go into verifying that single particle migration is valid during event catalog generation. At the basic level, the event must corresponding with the entity ID and sublattice. In addition, it is required that the destination location of the particle migration be currently "unoccupied", where unoccupied in this context means that the location cannot

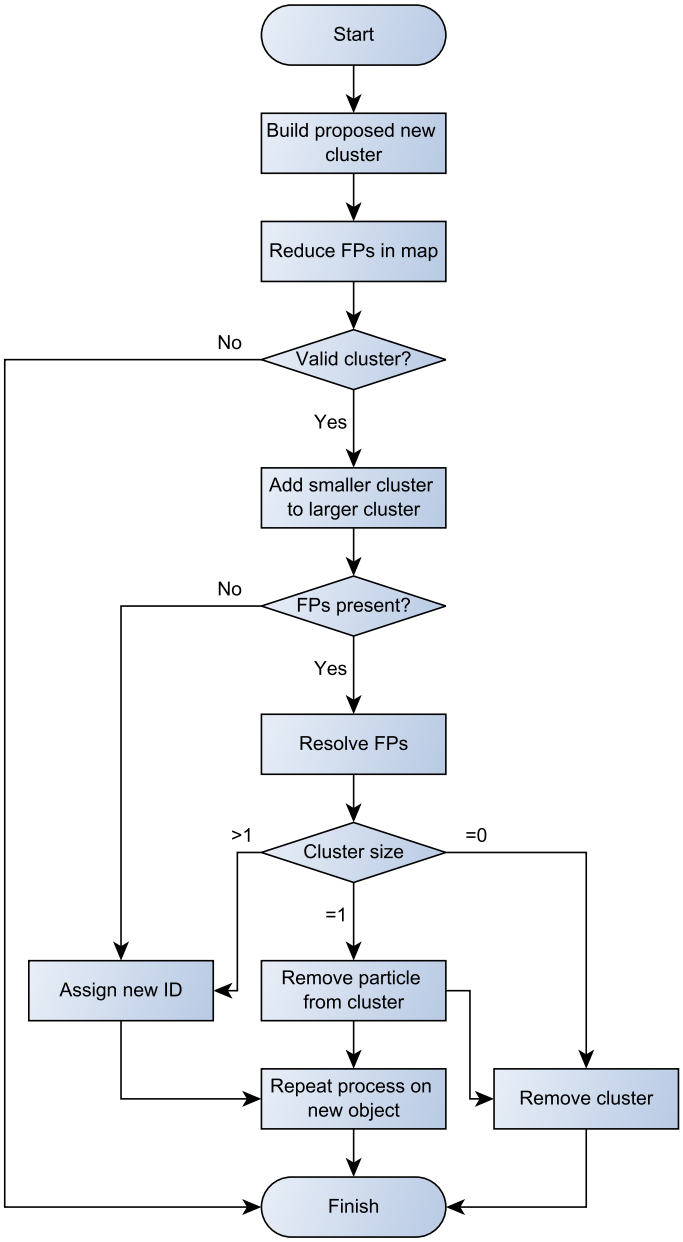



**Figure 3.9:** Cluster-cluster interactions algorithm flowchart.

contain a defect. This is an important distinction, as some defects such as vacancies only make sense on locations that would be considered occupied by a background lattice atom. This can be a time consuming step, as a naive approach would require iteration through the list of particles in the system and checking each location in turn. Additional bookkeeping structures were added to the code to reduce this lookup from linear time to constant time with respect to the particle count, significantly reducing the time spend on this step.

Finally, with the development of the cluster model, additional checks needed to be added. Single particle migration events are intended for single particles, and since the list of interacting objects in the system also contains cluster objects, these objects need to be screened out. In addition, as discussed earlier in section 3.3.2, intracluster migrations are not considered in the model, so particles that are currently members of cluster objects also must be screened out.

In this context, the rates in question are thermally activated migrations that follow an Arrhenius relation, with a migration rate given by [50]:

$$r_i = \nu_0 \exp\left(-\frac{E_m^i}{k_B T}\right) \tag{3.1}$$

where $r_i$ is the rate of migration for the event $i$, $\nu_0$ is the migration attempt frequency, $E_m^i$ is the migration energy for the event $i$, $k_B$ is the Boltzmann constant, and $T$ is the absolute temperature of the system. All of these parameters are given as input to the system; the migration energy and attempt frequency are given per event/particle, as they can vary between species and event, while the temperature and Boltzmann constant are given in the system configuration file. While the Boltzmann constant is a constant, it is still taken as input to allow for flexibility in the units used for energy and temperature.

Once all of the checks have passed, and acceptable single particle migration events have been added to the event catalog, carrying out the event if selected is fairly straightforward. The particle is properly moved to the designated destination position, then considered for interactions as described in section 3.3. If the particle crossed any of the periodic boundaries, this information is also stored so that proper distance calculations can be performed at the end of the simulation.

### 3.4.2 Cluster Migration

Cluster migration functions very similarly to single particle migration. Most of the same verification checks are performed, with the exception of the cluster detection rules. In this case, cluster migration events should only be performed on cluster objects, so single particles need to be screened out. It is also important to note that the cluster migration events can only be applied to cluster objects, and not the particles that constitute that cluster, which must also be screened out.

When a cluster migration event is selected and performed, the process proceeds similarly to that of single particle migration. The cluster object is moved to it's new designated position, and it's boundary crossing information is updated. In keeping with the cluster model, the individual particles that constitute the cluster are also moved to the same position, and their

boundary crossing information updated. Once the migration is complete, the cluster is checked for interactions with other objects in the system. It is important to note that only the cluster is checked for interactions, and not it's constituent elements, as they are considered to be part of the cluster, and are not able to interact with other objects on their own.

### 3.4.3 One-dimensional Migration

One-dimensional migration mechanisms for both single particles and clusters are based on their three-dimensional counterparts, but with addition checks on the particle direction. This type of migration is used for objects that move preferentially in one direction (forward or backward), than in other directions. This is primarily used for one- or two-dimensional objects such as dislocation loops or crowdions that move easier in one set of directions than others. For example, in the context of the iron system, SIA clusters can move easier along a given $\langle 111 \rangle$ direction, but requires more energy (crowdion rotation energy) to change to another $\langle 111 \rangle$ direction.

This mechanism is handled in the code by specifying two energy values for one-dimensional migration events in the action input file: the on-direction migration energy, and the off-direction migration energy. Additionally, all Entity objects in the system were modified to store their previous migration direction in order to facilitate this event type. When a particle or cluster is considered for one-dimensional migration, the process proceeds mostly the same as the three-dimensional process, but in this case, the migration direction is compared to the previous direction stored with the migrating entity. If the migration is in either the same direction or opposite direction, the on-direction migration energy is used to calculate the event rate (and as a result, event probability). If the migration direction is any other direction, the off-direction migration energy is used instead. The on-direction migration energy is also used if the migration event is the first migration that the particle will perform, as the particle has no previous migration direction to reference, and is therefore assumed to have no preference in direction until after its first movement. This is shown schematically in fig. 3.10.

### 3.4.4 Cluster Dissociation

Cluster dissociation events are applicable only to the single particles that are currently constituents of a cluster, while all other entities in the system are screened out. Dissociation events are considered on a per-particle basis, so in a 50 vacancy cluster, for example, there will be 50 possible dissociation events; one per constituent particle. Single particles are able to

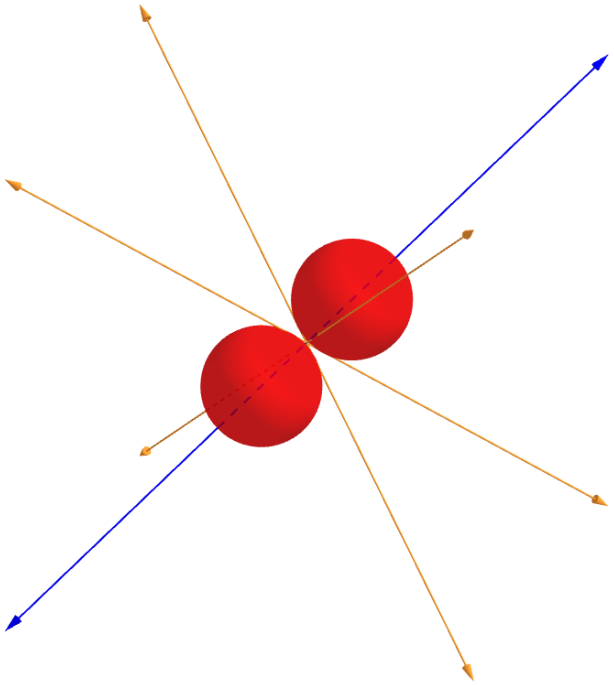


**Figure 3.10:** Schematic representation of one-dimensional migration mechanism. Defect shape results in lower energy migration along one set of directions (blue) than other directions (orange). The process is the same for both single particle and cluster migration.

dissociate from clusters based on the same Arrhenius relationship as migration, except with the dissociation activation energy $E_a$ substituted in place of the migration energy:

$$r_i = \nu_0 \exp\left(-\frac{E_a^i}{k_B T}\right) \tag{3.2}$$

where $E_a = E_b + E_m$ and is the combination of the dissociating particle's binding energy and migration energy. Unlike with the migration-type events, dissociation actions don't have direction specified. Instead, when a dissociation event is selected for execution, a new random location is generated such that it is on the correct sublattice, unoccupied (as with migration), and sufficiently far away from the center of the source cluster to prevent immediate recombination. The minimum dissociation distance is set by the interaction distance of the two particles, and the maximum dissociation distance is set to be lattice parameter beyond the minimum distance. This is shown schematically in fig. 3.11.

Unlike with single object migrations, cluster dissociation will result in two new unresolved objects: the dissociated particle and the remaining cluster object. As with the particle-cluster and cluster-cluster interactions, if a particle dissociating from a cluster results in a singleton cluster, the particle is removed from the cluster and the empty cluster is removed from the

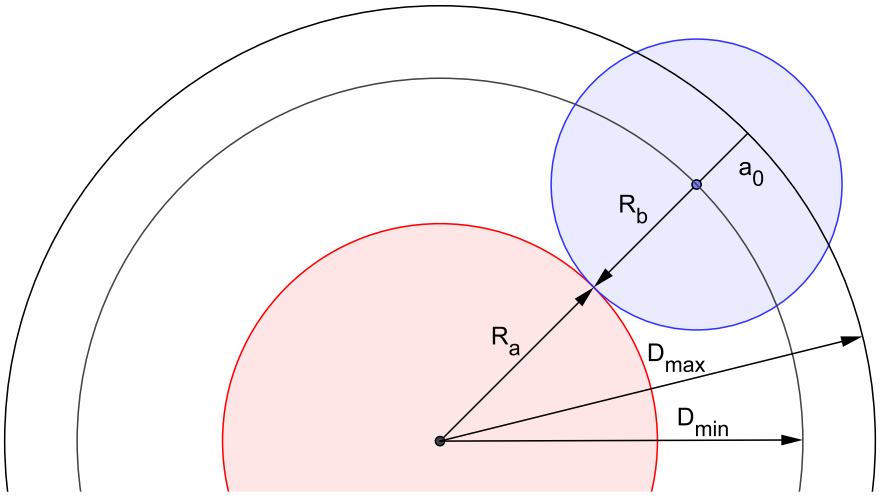



**Figure 3.11:** Schematic representation of cluster dissociation mechanism. The source cluster interaction region is indicated by the red circle, and the dissociated particle interaction region is indicated by the blue circle. The minimum dissociation distance is given by $D_{min} = R_a + R_b$, and the maximum dissociation distance is given by $D_{max} = D_{min} + a_0$.

system. Finally, both the dissociated particle and the remaining cluster/particle are checked for interactions with other objects in the system.

### 3.4.5 Defect Production

Defect production in the code is setup to allow the creation of any collection of particles or Frenkel pairs at a set rate. Unlike the other events that follow an Arrhenius relation, production rates are fixed with respect to temperature. Production events also need to be handled carefully within the main KMC algorithm, as until the event occurs, there is no entity attached to the event. Since events are generated by comparing entities with actions and finding compatible pairs, production events would never be chosen, as they do not have a corresponding entity. To handle this, a list of production type events is generated when the action information is read in, and this list of events is always added to the event catalog during generation.

When a production event is selected for execution, a new random position and particle are generated using the same mechanism as the initial population of the system. Once the particle is added it is then checked for interactions with other objects in the system. Finally, if the generated particle has a Frenkel pair associated with it, the Frenkel pair particle is generated and checked for interactions in the same manner as the original particle.

## 3.5 Code Efficiency

Several algorithmic modifications to the original code were required in order to effectively simulate the systems of interest in this study. While there were a number of minor improvements made as a result of development, the most significant are described here.

### 3.5.1 Action Set Mapping

One major efficiency factor was the algorithm used to populate the event catalog from the Entities and Actions available in the system. In the original code, the event catalog was populated by iterating over both the Entities list and Actions list, comparing each of them pairwise, and selecting the events that were acceptable. In the original code application (oxygen diffusion in lanthanum-doped ceria [46]), there was only one mobile species (oxygen vacancies), so when the vacancies entities in the system were checked against oxygen actions, they were almost all acceptable. There were some combinations rejected for other check reasons (e.g., destination occupied, on wrong sublattice, etc.), but for entity-action compatibility, they were all valid.

With the addition of the cluster model and the large number of species that result from such a model, the naive algorithm from the original code would no longer be acceptable. In this study, there would be large numbers of possible actions for cluster types that may not even exist in the system. For example, if vacancy clusters up to $V_{50}$ were permitted and mobile, but only one vacancy existed in the system, that vacancy would be compared against all of the cluster migration events for $V_2$ through $V_{50}$, even though none of those actions would be possible with the given system composition. This is an exaggerated example, but the problem remains: there will be a large number of possible events specified for the system that cannot occur without the required populations formed, yet will still be validated against for every particle in the system during catalog construction.

To solve this problem, an additional bookkeeping structure was added to the code to ensure that entities are only compared against actions that they can potential perform. This is implemented in the code using a standard map structure to map Entity type to a list of Actions that relate to that action. When the action information is read in from the input file, the actions are split into lists by Entity type, rather than all being placed in one large list. This structure is shown schematically in fig. 3.12. Later, when the event catalog is being constructed, rather than comparing each Entity against the full list of possible Actions in the system, it is only compared against the mapped list that contains only Actions that the Entity of interest can possibly perform. This process is shown schematically in fig. 3.13.

To demonstrate the efficiency improvement, a test system was setup and evolved, and the

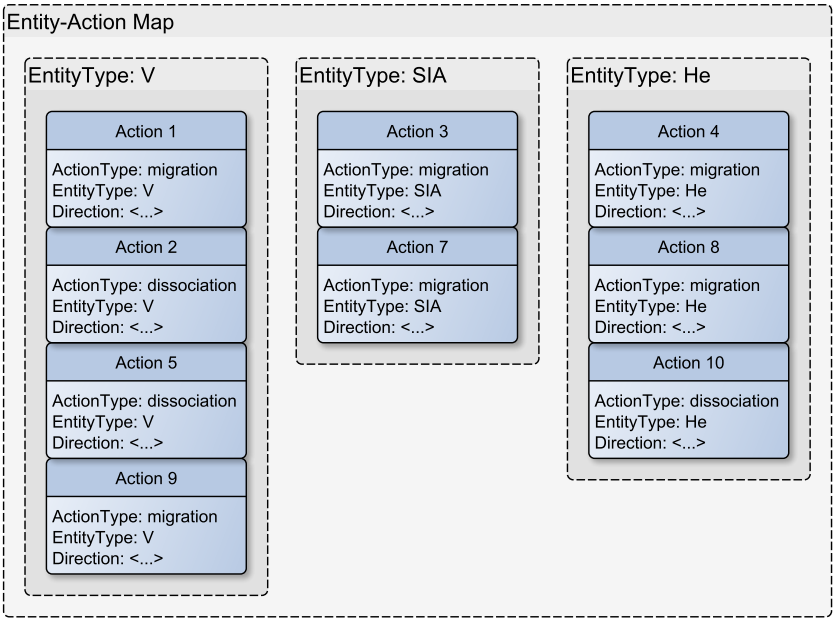
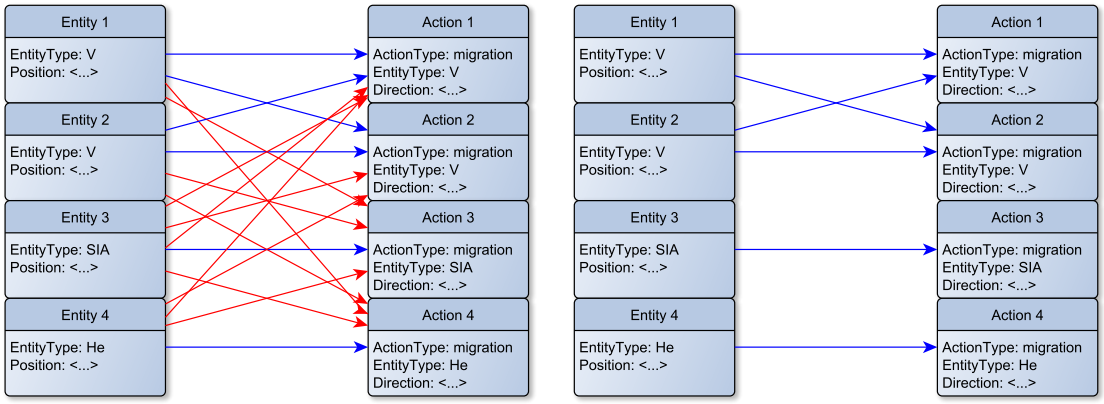



**Figure 3.12:** Schematic representation action set mapping. Entity types are mapped to lists of Actions that contain only Actions that Entities of said type can potentially perform.

simulation run times compared. The test system used was one of the potential systems to be used in the application study. This system consists of a $125 \times 125 \times 125$ unit cell BCC iron system, at 300 °C, with initial populations of 703 I-V Frenkel pairs, and 72 interstitial helium atoms. Single particle and cluster migrations were added for iron SIAs and vacancies, and He interstitials. In order to demonstrate the effects of the algorithmic change, acceptable clusters were limited to interstitial clusters up to $I_{25}$, and vacancy-helium clusters of combinations up to $He_{10}V_n$, where $n$ was set to values of 5 through 60, in steps of 5. As the objective was simulation run time and not any material property, the simulations were run for 100 000 steps, with each setup run 10 times to generate statistics. The results of these simulations are shown in fig. 3.14. The simulation time goes almost as the number of actions in the system, and as such, the mapped algorithm displays significantly improved linear time behavior. At first glance it may seem like this should result in linear behavior only to a point, follows by constant-time above some threshold. For example, if clusters never form (for physical reasons) beyond $V_{20}$, then one might expect linear behavior up to $V_{20}$, followed by constant time beyond that, as these clusters should never be compared with anything. The complete linear behavior is a result of the way particle dissociation is considered. As dissociation is essentially particle migration with an increased migration energy, the event is considered based on the dissociating particle, rather than the cluster from which the particle is dissociating. For example, a $V_{20} \longrightarrow V_{19} + V$ event is considered a V event, not a $V_{20}$ event. As a result, from the previous example, even if there




**(a)** Original action lookups. **(b)** Mapped action lookups.

**Figure 3.13:** Schematic representation of action lookups during event catalog construction. Blue arrow represent potential combinations, red arrows represent impossible combinations. In the original algorithm (a), each Entity is compared against all possible Actions. In the new algorithm (b), Entities are only compared with lists of Actions that said Entities can potentially perform.

are no $V_{20+}$ clusters, the vacancy dissociation event from those clusters will still be considered, resulting in improved but still linear time.

### 3.5.2 OpenMP Parallelization

A second major improvement in code run efficiency was the introduction of OpenMP to speed up time-consuming parts of the algorithm. OpenMP is a shared memory parallel processing API that is implemented through a set of compiler directives, environment variables, and a runtime library. OpenMP compiler directives are supported by most major compilers, so this should not affect the portability and deployment of the code. Since the code is written in object oriented style and most of the data structures and optimizations are based on object references (pointers), the shared memory paradigm of OpenMP is ideal for parallelization. In shared memory parallelization, all parallel threads share the same memory space (as the name implies), so there is no need to manually keep data consistent between threads. Unfortunately, while this does make parallelization of the code easier, it does limit the extend to which the parallelization can be utilized. As the parallelization is shared memory, it can only be expanded to use as many cores and the system is able to share memory between. In most cases, this means that the number of processes is limited to the number of cores on a given node, however if specialized hardware is present, shared memory between nodes is possible.

All variables within an OpenMP (OMP) parallel region are shared by default, but variables can be specifically declared as private if necessary. This makes parallelization of this code

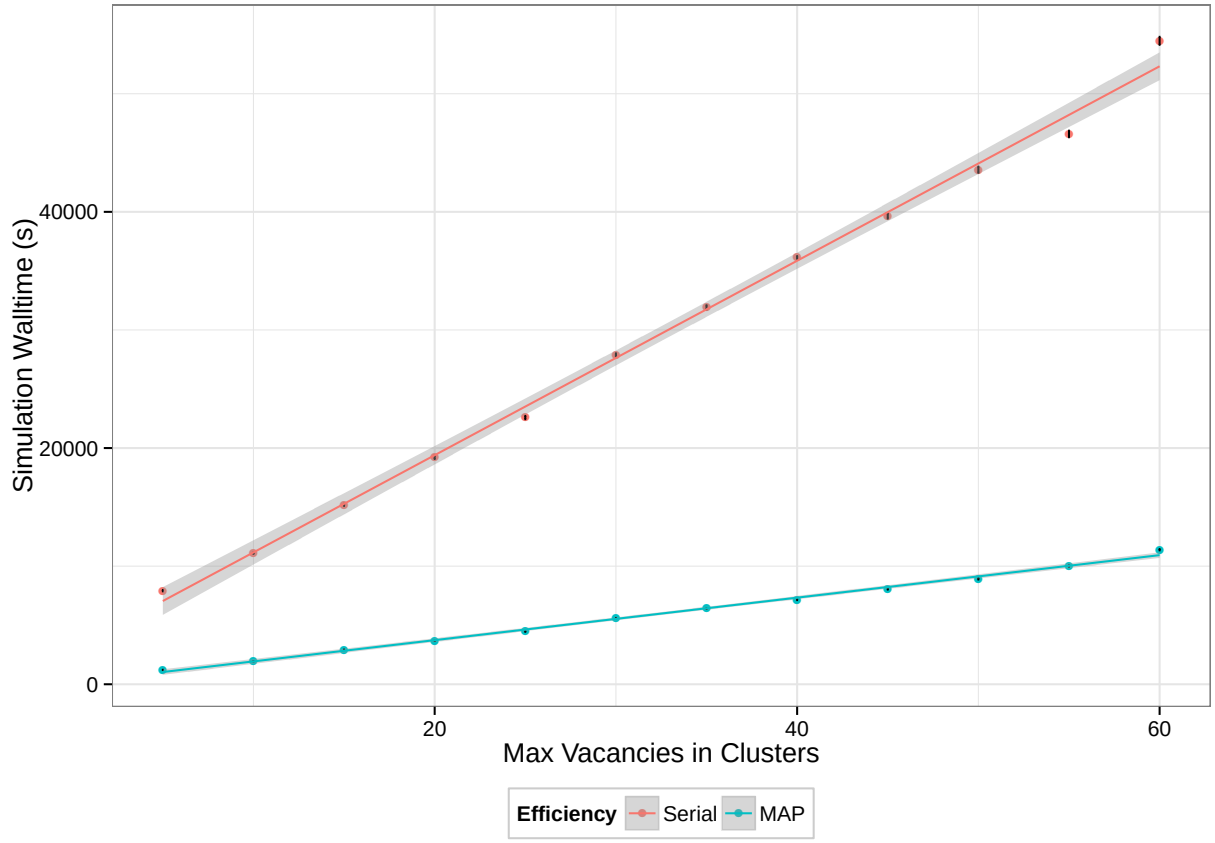
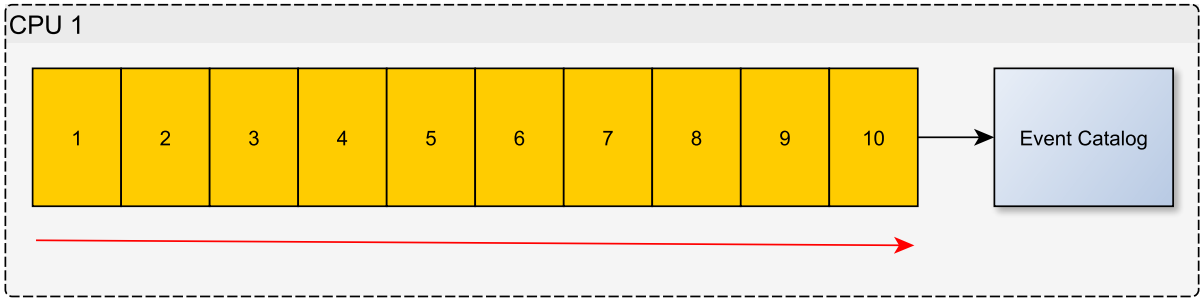


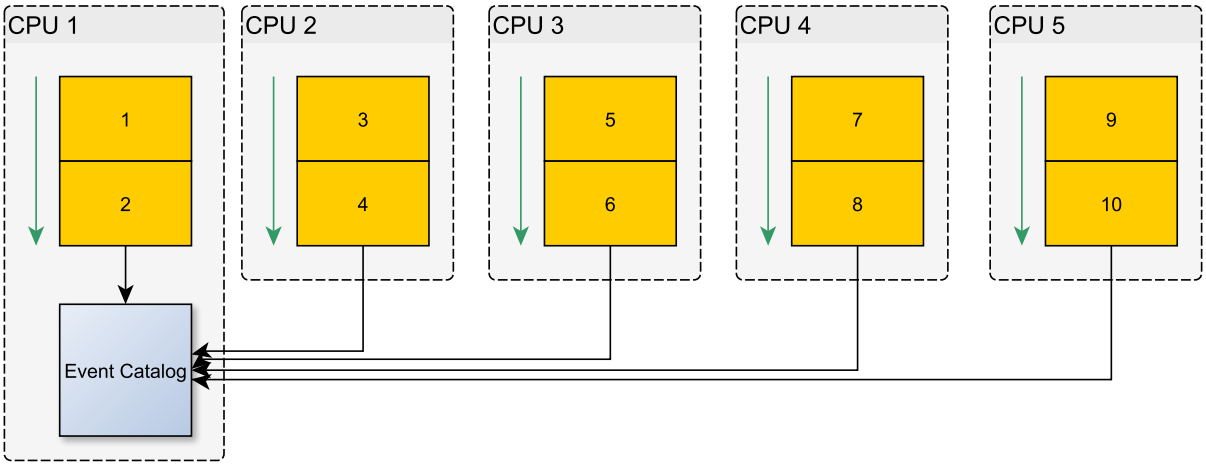

**Figure 3.14:** Comparison of simulation runtime vs. number of permitted vacancy clusters. The simulation time goes almost as the number of actions in the system, and as such, the mapped algorithm displays significantly improved time behavior when compared to the original linear algorithm.

relatively straightforward in primarily read-only situations. In this code, the most computational expensive step is building the event catalog, as this requires that every entity in the system be iterated over and compared with all possibly applicable actions (see section 3.5.1). This is, for the most part, a read-only operation, as none of the Entities in the entity list change during the construction of the event catalog. This only thing that does change is the event catalog that is being constructed. As a result, this can be parallelized by splitting the list of Entities evenly between OMP cores, constructing partial event catalogs for the Entities privately on each core, and then combining them into the single, full event catalog at the end. This is shown schematically in fig. 3.15.

There are several points that need to be considered when implementing this in practice. First, OpenMP works on loops by determining the number of loop iterations that will be performed, dividing these iterations between the available OMP cores, an then entering the loop region. This creates a conflict with the implementation of the code, as the Entities in the system are




**(a)** Serial event catalog construction.




**(b)** Parallel event catalog construction.

**Figure 3.15:** Schematic representation event catalog construction. Yellow boxes represent entity-action comparison workload units. In the serial algorithm (a), all comparison work is done in series on a single core. In the parallel algorithm (b), comparison work is split up between available cores and carried out in parallel.

stored as an STL list object. This is ideal for this type of system, as it allows Entities to be added or removed from the system quickly without causing fragmentation or reallocation with growth, as would occur if done with an STL vector. Unfortunately, the trade of on the decision to use a list over a vector is lack of random access support that vectors possess. Unlike vectors, which allow any element of the vector to be accessed by its index, lists must be iterated from beginning to end. This is a problem for OpenMP, as the workload divider needs to know the number of loop iterations to split, and the loop body needs to be able to access whichever iterations it is assigned by the workload divider. In order to facilitate this, a copy of the Entity list is created in vector form, which can then be used by OpenMP. As mentioned earlier, since the Entity list does not contain Entity objects, but rather references to Entity objects, it can be copied quickly without a significant memory cost. This vector is only used during the event

catalog construction, and is removed after the list is constructed. This is important as the vector will become inconsistent as soon as an event is selected and performed.

The second major point of concern is the reduction of the privately constructed partial event catalogs into a single event catalog. STL lists are not thread-safe, meaning they cannot be reliably modified by multiple threads concurrently, which is likely to occur in this context. This means that the event catalog cannot be populated by all OMP threads simultaneously, and instead requires that individual threads construct partial lists and combine (reduce) them at the end. OpenMP does support automatic reduction of specific variables, however currently only for language primitives (e.g., int, float, etc.). Since the object being reduced in this case is an STL list, the OpenMP automatic reduction direction cannot be used. As a result, reduction of the list is performed manually. Prior to entering the OMP parallel region, an empty partial list is created. When entering the parallel region, this list is declared as private, so that each core will only attempt to modify a list to which it has exclusive access. Once the parallel loop blocks have executed, the partial event catalogs can be safely combined into the main single event catalog by enclosing the list append operation in a "critical" section, which guarantees that the contents of the region will only be executed by one thread at a time. While it would be possible to construct the list in-place by enclosing the list append operation in a critical section, there is overhead and thread blocking involved with such sections, so they should be used as little as possible. This will only happen once per core in the partial population scheme, but would occur for every valid event in the in-place scheme, which might even it slower than the serial algorithm.

To demonstrate the parallel efficiency improvement, a test system was setup and evolved, and the simulation run times compared. The test system used was one of the potential systems to be used in the application study. This system consists of a $125 \times 125 \times 125$ unit cell BCC iron system, at 300 °C, with initial populations of 703 I-V Frenkel pairs, and 72 interstitial helium atoms. Single particle and cluster migrations were added for iron SIAs and vacancies, and He interstitials. In order to demonstrate the effects of the algorithmic change, acceptable clusters were limited to interstitial clusters up to $I_{25}$, and vacancy-helium clusters of combinations up to $He_{10}V_{25}$. The number of OpenMP cores was allowed to vary from 1 to 12 cores, which was the maximum number available per node on the cluster used for the study. As the objective was to study simulation run time and not any material property, the simulations were run for 1 000 000 steps, with each setup run 10 times to generate statistics. The results of these simulations are shown in fig. 3.16.

These results show significant improvement in runtime with lower numbers of cores, with some diminishing returns as the core count increases. There are a number of possible causes for this result. For one, there is some overhead involved with the parallelization process. Since the

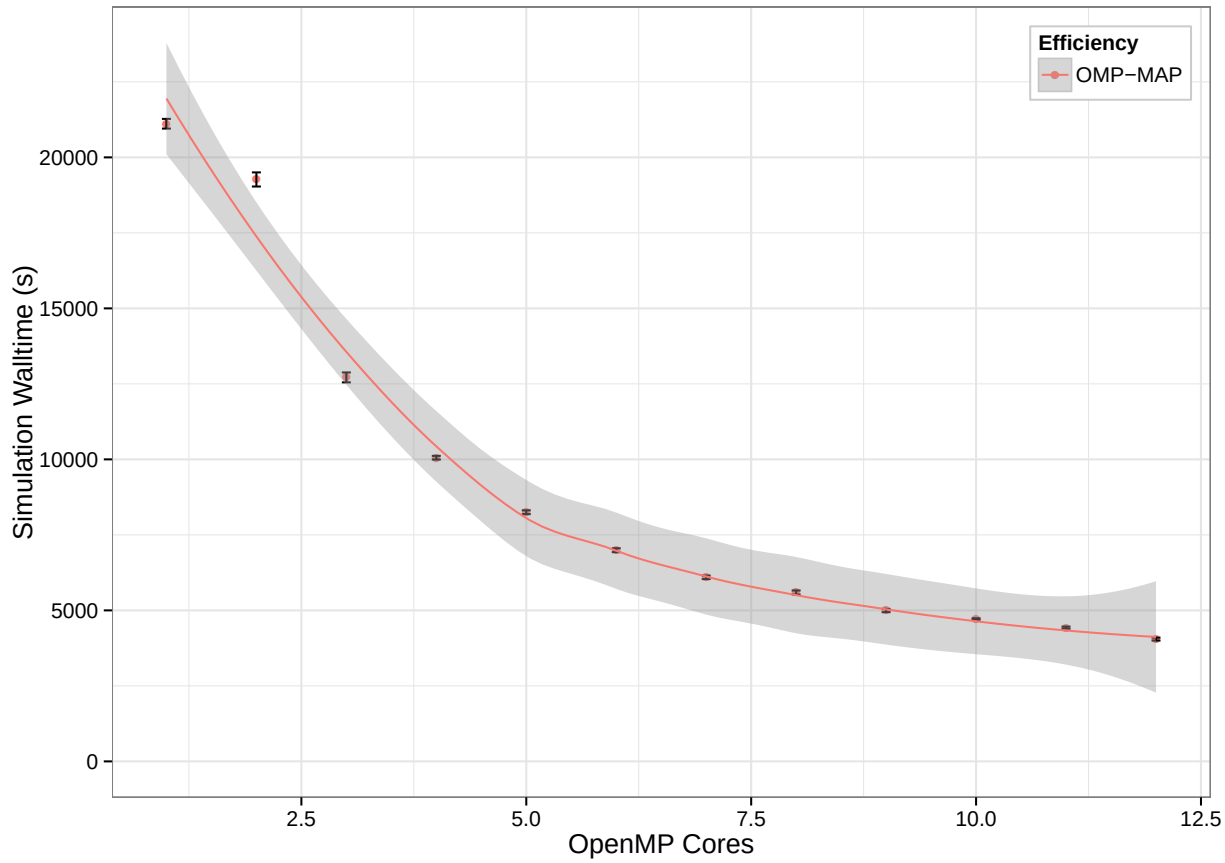



**Figure 3.16:** Comparison of simulation runtime vs. the number of OpenMP cores used during simulation.

parallelization is based on the number of entities in the system, there will be a bottleneck with a low number of particles where the parallelization overhead dominates the speed improvements from increased cores. Second, as a result of the way the entity list is manipulated throughout the code, the list is stored as an STL list. When doing the event catalog construction, the entity list must be split between cores, processed into one event catalog per core, then combined into a single catalog at the end. In order to do this thread-safely in the current version of OpenMP, the entity list must first be copied into an STL vector before processing, and the partial event catalogs must be reduced manually. These are not major inconveniences, but do negatively impact performance. Future versions of OpenMP are reported to introduce functionality to improve these issues, so this issue will be revisited when the new version is available.

Overall, the combination of both the action mapping mechanism and OMP parallelism significantly improves the speed of the code. A comparison of runtimes vs. acceptable vacancy size clusters (similar to fig. 3.14) is given in fig. 3.17. The results show clear improvement, as the combined map/OpenMP implementation almost constant-time behavior when compared to the

original linear algorithm. The runtimes of simulations with varying numbers of particles were also compared using different OpenMP core allocations. These results are given in fig. 3.18, and show clear improvement in simulation runtime as the number of OpenMP cores is increased.

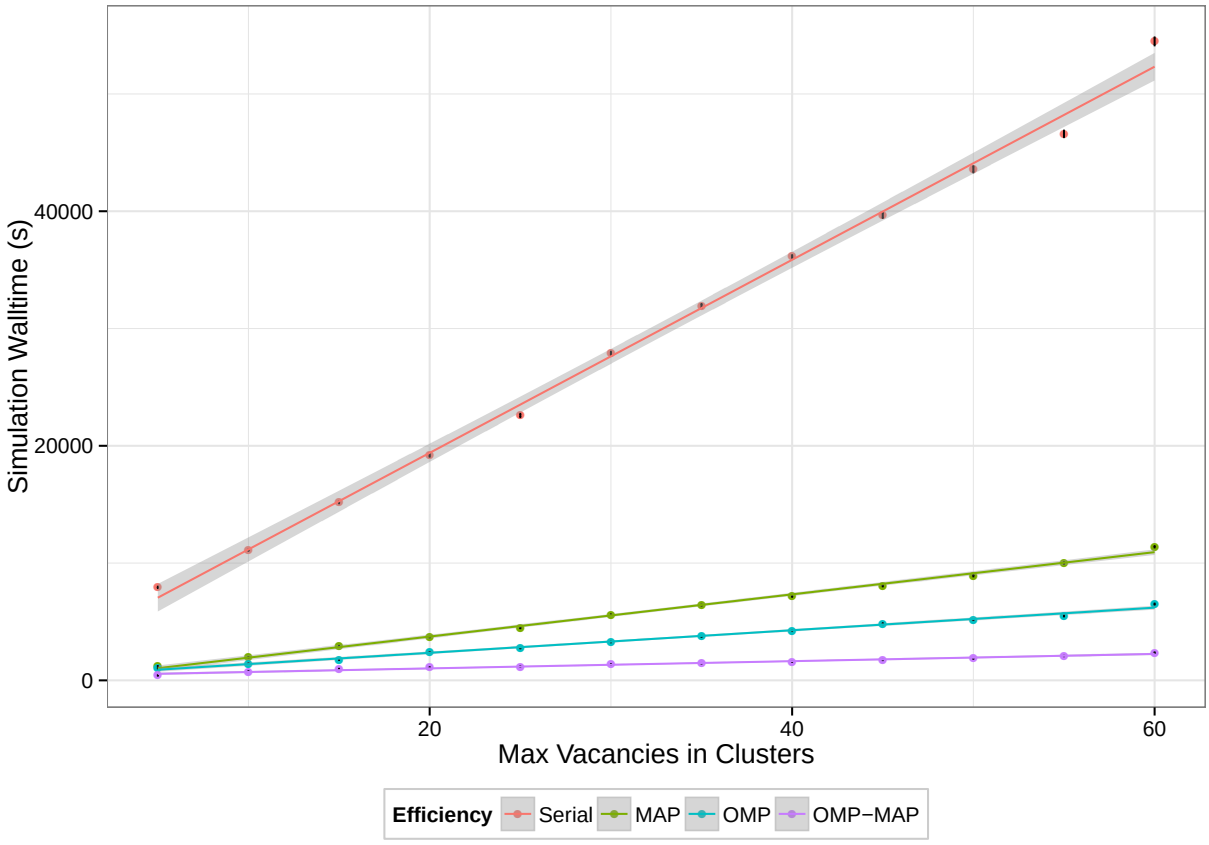



**Figure 3.17:** Comparison of simulation runtime vs. number of permitted vacancy clusters for combinations of efficiency improvements. The number of events in the system goes almost as the number of actions in the system, and as such, the combined map/OpenMP implementation is almost constant-time behavior when compared to the original linear algorithm.

### 3.5.3 Population Tracking

Another major change was the introduction of population tracking. In the original code and its application, there were a fixed number of particles in the system, and they did not transform from one species to another. As a result, there was no need population tracking at that point. To satisfy this need initially, the code would loop through the entire list of entities and tabulate a population distribute at each time series output step. This worked perfectly well for its function, but it resulted in time series output steps being significantly slower than they needed to be.

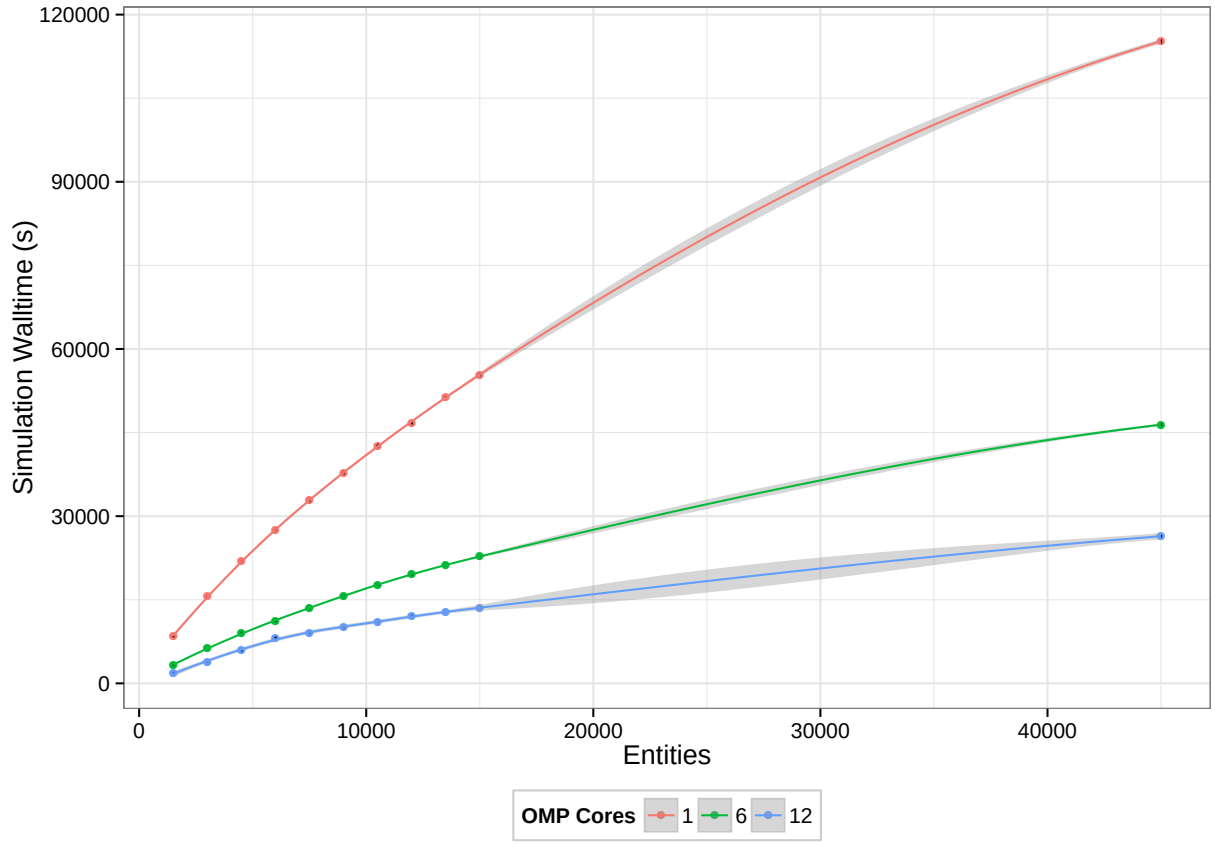



**Figure 3.18:** Comparison of simulation runtime vs. number of particles in the system for different OpenMP core allocations.

This becomes more significant as the number of particles in the system increases, or as the number of time series output steps increases. In the new code and the application presented in this work, the systems contain large numbers of particles, and the population distribution was the primary result of interest, which meant the code needed to be improved.

To improve the efficiency of determining population distributions, the code was rewritten so that particle populations are tracked from the beginning of the simulation. Whenever particles are added or removed from the system, the corresponding populations are updated. In addition, whenever clustering interactions result in the transformation of particle types, the populations of the types involved are also updated. In this way, any changes to the tabulated populations occur whenever changed in the actual populations occur, so the population distribution is always up to date. This completely eliminates the need to tabulate a full population distribution, and while depending on the number of time series output steps, this may not result in enormous runtime changes, any change to avoid iterating the entire entity list is an important improvement.

## 3.6 Resume Functionality

A final significant change was the introduction of the ability to resume runs. This was introduced as a way to work around the walltime limits imposed on the computational cluster used for the study. The longer runs needed more time to run then single jobs were permitted, so this functionality was introduced so that the state of the system that reached the walltime limit could be resumed in a subsequent job. This functionality also allows runs to be continued if additional evolution is required. For example, if a system was evolved to some time and then analyzed, and it is determined that some phenomena of interest is still in process, the system can be resumed with a new end condition and evolved until the desired phenomena is complete.

This functionality is implemented in the code by periodically saving the state of the system to an external restart file. The key system evolution information includes the current timestep, current evolution time, and current DPA level. After this information is recorded, the types and positions of every individual particle is output. The particles that are members of clusters are output as single particles, however since all elements of a cluster have the same position, the will immediately reform a cluster when the state is reloaded. This save process is performed at a fixed KMC step interval, which can be changed as appropriate. This file stores only the state of the system, not any of the configuration information, so the original input files are still required to resume a run.

In order to minimize the work necessary to restart a run that had exceeded its walltime limit, the code is set to read a command line switch to determine if it should attempt to resume a run. If this switch is set, the code will read the configuration file as before, but will load the system state information from the resume file instead of initializing the system at zero and creating the initial distribution that would normally come from the populations specified in the configuration file. At this point, the code will proceed through its algorithm as it normally would. If the run is being resumed as a result premature termination, the only change necessary is the command line switch in the job file. If the system is being continued to a new end condition, the only additional change is to set the new end condition in the configuration file before restarting.

# Chapter 4

# KMC Simulations

In addition to the accelerated time evolution mentioned previously, Kinetic Monte Carlo's other major benefit is the ability to study the effects of different system parameters independently. In this study, the effects to be study are the cluster radius models, the cluster dissociation energy models, and the cluster mobility models. Each of these are specified as input to the KMC code, and as such, can be varied and combined as desired to study their effects. The systems of interest are studied first under annealing conditions (i.e., no damage) in section 4.5, followed by damage simulations in section 4.6.

## 4.1 General Simulation Parameters

In this model, defects are considered as objects characterized by their space coordinates, type (I, V, He, $I_n$, $V_n$, and $He_mV_n$ in this study), mobility, and dissociation rate. Most of the simulations in this section used the same set of assumptions and system settings, with variations primarily in the three models mentioned above. The assumptions for vacancies, interstitials, and helium dopants, as well as the system configuration and run conditions, are described here.

### 4.1.1 Vacancy Properties

For single vacancies, the migration energy $E_{m,V}$ was taken to be 0.69 eV, as found with EAM potentials [52]. This is close to the ab initio value of 0.65 eV [26] and the experimental value of 0.55 eV [58]). For vacancy clusters ($V_{n>1}$), the defects are assumed to be mobile, but with mobility decreasing with size [52, 59]. In this case, the migration rate decreases with cluster size by applying a geometric progression of common ratio $p^{-1}$ to the migration frequency $\nu_0$ while keeping the migration energy $E_m$ constant. Thus, the migration rate is given similar to the single particle rate (eq. (3.1)), but with an attempt frequency that varies with cluster size:

$$r_{V_n} = \nu_V(n) \exp\left(-\frac{E_{m,V}}{k_B T}\right) \tag{4.1}$$

$$\nu_V(n) = \nu_0 \left(p^{-1}\right)^{n-2} \qquad n \geq 2 \tag{4.2}$$

The base attempt frequency $\nu_0$ was the same for both single and cluster vacancies, and was taken to be $6.0 \times 10^{12}$ Hz, which provided the correct self-diffusion coefficient for iron [52]. The progression factor $p$ was taken from the same study to be $p = 100$. The parameter choices adopted in that study to describe cluster diffusivity were based on MD studies of iron and are consistent with KMC models already published [53, 55, 60–62].

### 4.1.2 Interstitial Properties

The self-interstitial atom (SIA) in iron is known to have a $\langle 1\,1\,0 \rangle$ dumbbell configuration in its most stable configuration [63, 64]. For single iron interstitials, the migration energy $E_{m,I}$ was taken to be 0.04 eV. This value is much lower than the experimental value of 0.3 eV [63], but is the value typically found in MD simulations [61]. SIA clusters, on the other hand, are described as a collection of $\langle 1\,1\,1 \rangle$ crowdions [60, 65–67]. These MD simulations show that both single SIA and SIA cluster mobility is based on glides along $\langle 1\,1\,1 \rangle$ directions. For single SIAs, the rotation energy from $\langle 1\,1\,0 \rangle$ to $\langle 1\,1\,1 \rangle$ directions is relatively low (~0.1-0.3 eV), and can also easily change $\langle 1\,1\,1 \rangle$ directions, resulting in almost fully 3D motion, depending on temperature. For SIA clusters, the probability of direction change reduces with increasing size [68]. This model is implemented using the one-dimensional migration mechanism in section 3.4.3, where the on-direction energy is the normal migration energy, and the off-direction energy contains the additional energy required for crowdion rotation, $E_{a,rot}$.

In this study, single SIAs were assumed to have full 3D motion at all temperatures ($E_{a,rot} = 0$), as is typically done in KMC simulations [53, 55, 60, 62]. In contract, SIA clusters were assumed to be constrained to a 1D $\langle 1\,1\,1 \rangle$ random walk (i.e., $E_{a,rot} = \infty$) [52]. In this model, the cluster is initially allowed to move in all 3D directions with the same energy, and once it has moved once, it is constrained to move in that direction (or the reverse direction) for the remainder of the simulation. This was implemented by setting the off-direction migration energy sufficiently high so that off-direction migration will never be selected.

Similar to the vacancy cluster mobility model described in section 4.1.1, the migration energy of an SIA cluster was taken to be constant, with the attempt frequency that decreases as a function of the cluster's size. Unlike the vacancy clusters, the SIA cluster mobility was set to decrease according to a power law ($n^{-s}$). MD results by Osetsky [65] suggest a law close to $1/\sqrt{n}$ ($s = 0.51$), which was used in this study:

$$r_{I_n} = \nu_I(n) \exp\left(-\frac{E_{m,I}}{k_B T}\right) \tag{4.3}$$

$$\nu_I(n) = \nu_0 n^{-s} \tag{4.4}$$

The base attempt frequency $\nu_0$ was the same for both single and cluster SIAs, and was the same value used for vacancies ($6.0 \times 10^{12}$ Hz).

### 4.1.3 Helium Properties

As discussed in section 2.2, simulations done with more recent interatomic potentials show helium as preferring the tetrahedral interstitial sites over the octahedral sites, however their formation energies are similar. To simplify the simulations in this study, helium was assumed to occupy the octahedral interstitial sites. As will be seen when the cluster radius models are discussed in section 4.2, the distance between an octahedral site and neighboring tetrahedral sites (~0.25 Å) is small compared to the interaction distances used (>3.3 Å), so this assumption should not significantly influence the results of the simulation. The migration attempt frequency was taken to be the same as that of iron ($\nu_{0,He} = 6.0 \times 10^{12}$ Hz), while the migration energy $E_{m,He}$ was taken from MD simulations to be 0.078 eV [27]. While not explicitly stated, it is also common in other KMC simulations to assume that helium will not cluster with itself, and will only form clusters with vacancies ($He_mV_n$ bubbles).

### 4.1.4 Simulation System

The majority of the annealing runs were based on the same configuration. The simulation system was a $125 \times 125 \times 125$ BCC iron unit cell system with periodic boundary conditions. The temperature was set to 300 °C (~$0.3T_m$), and initial populations were randomly distributed throughout the system. 2000 vacancies (~0.05 at%) were introduced in all simulations, which was chosen based on the initial conditions of one of the experimental irradiation setups. Helium was also introduced at varying concentrations of 1 %, 5 % and 10 % of the vacancy concentration (also written as He:V ratios of 1:100, 1:20, and 1:10, respectively) in order to study their effect on the various models described below. These simulations were originally run for $4 \times 10^6$ KMC steps, but were extended in many cases (typically in increments of $4 \times 10^6$ steps) to allow for proper comparison at real time steps. Each configuration was run with 10 different random initial configurations to obtain statistics. Distribution results are typically reported as averages

with accompanying standard error bars. Time series evolutions are typically reported based on a single run as including the details of all averaged runs would make the plot unreadable.

## 4.2 Cluster Interaction Radius Models

As part of the clustering model discussed in section 3.3, every defect in the system is described as a point particle with some interaction radius. However, the choice of radius is a topic of discussion. To compare the effect of the defect interaction radius on cluster formation, simulations were performed using four different interaction radius models.

### 4.2.1 Constant Radius Model

The first model is the one used by Deo and Okuniewski [11, 50], and assumes that all defects have the same interaction radius, and that only particles within one unit cell interact. Since interactions are based on the distance between two objects, the constant interaction radius of each particle is given by half that distance:

$$R = 0.5a_0 \tag{4.5}$$

where $a_0$ is the lattice parameter of the system.

### 4.2.2 Vacancy-Dependent Model

The second model (by Ortiz et al. [51]), frequently referred to as the "$R(V)$" model, allows the interaction radius to grow with increasing vacancy cluster size, but independently of the concentration of helium in the cluster:

$$R_{He_mV_n,I_n} = Z_{I,V}\left(\sqrt[3]{\left(\frac{3n\Omega_{Fe}}{4\pi}\right)} + 1.15a_0\right) \tag{4.6}$$

where $R$ is the interaction radius of a vacancy/interstitial cluster containing $n$ defects, $\Omega$ is the atomic volume, and the $1.15a_0$ term is calibrated such that the I-V recombination distance is $3.3\,a_0$, in agreement with previous works [69, 70]. The biasing factors $Z_{I,V}$ are introduced to account for the fact that SIA-type defects have larger strain fields than vacancy-type defects, and are taken to be $Z_V = 1$ and $Z_I = 1.15$. These values are determined from experimental results and are a common choice in this type of model [52, 53, 60]

### 4.2.3 Helium Addition Model

The third model, frequently referred to as "$R(V+He)$", is very similar to the second model with regard to vacancy and interstitial clusters, but differs in the treatment of $He_mV_n$ clusters. This model is based on one used by Domain et. al. [52] for Fe-Cu simulations and assumes that any increase in particles in the cluster increases its interaction radius:

$$R_{V_nHe_m,I_n} = Z_{I,V}\left(\sqrt[3]{\left(\frac{3\left(n\Omega_{Fe}+m\Omega_{He}\right)}{4\pi}\right)}+1.15a_0\right) \tag{4.7}$$

Here, the effective volume of the defect is given by the quantity $(n\Omega_{Fe}+m\Omega_{He})$, which results in an interaction volume that increases with both the number of vacancies and the number of heliums in the bubble. In the absence of helium (i.e., $V_n$ and $I_n$ clusters), this model reduces to the vacancy-dependent model in eq. (4.6).

### 4.2.4 Helium Equilibrium Model

The forth model, frequently referred to as "$R(V-He)$", is also similar to the second model with regard to vacancy and interstitial clusters, but again differs in the treatment of $He_mV_n$ clusters. This model assumes that addition of helium reduces the strain field created from the void, so any increase in helium in the cluster reduces its interaction radius:

$$R_{V_nHe_m,I_n} = Z_{I,V}\left(\sqrt[3]{\left(\frac{3\left|n\Omega_{Fe}-m\Omega_{He}\right|}{4\pi}\right)}+1.15a_0\right) \tag{4.8}$$

Here, the effective volume of the defect is given by the quantity $|n\Omega_{Fe}-m\Omega_{He}|$, representing deviation from an equilibrium, which results in an interaction volume that decreases with an increasing number of heliums in the bubble. In the absence of helium (i.e., $V_n$ and $I_n$ clusters), this model reduces to the vacancy-dependent model in eq. (4.6).

## 4.3 Cluster Dissociation Energy Models

As discussed in section 4.1, all defects are also characterized by their dissociation energy from clusters. There are two dissociation energy models considered in this study.

### 4.3.1 Constant Dissociation Model

The first model is the one used by Deo and Okuniewski [11, 50], and assumes that each defect type has a constant dissociation energy. SIAs are assumed not to dissociate from SIA clusters, and vacancy dissociation energies vary only by the presence of helium in the cluster (i.e., $He_mV_n$ vs. $V_n$). Helium will only dissociate from $He_mV_n$ clusters, and dissociates with energy $E_{d,He} = 2.078\,\text{eV}$. Vacancies will dissociate from $V_n$ voids with energy 0.89 eV, and from $He_mV_n$ bubbles with energy 2.69 eV.

### 4.3.2 Variable Dissociation Model

The second model is based on dissociation energies calculated by Lucas [33]. These energies are shown in fig. 4.1, and vary primarily with He/V ratio and number of vacancies in the cluster. To use these energies in the simulation, the given data values were extracted and interpolated as a function of ratio for each given number of vacancies. As the energies for both helium and vacancy dissociation appear to saturate as the number of vacancies in the cluster increases, the values for the highest vacancy count were applied to all larger clusters. When the dissociation energies were calculated for a given cluster, larger clusters were mapped down to the largest reported vacancy size if necessary, the He/V ratio was calculated, and the dissociation energy was evaluated from the interpolated spline corresponding to the cluster's vacancy population. In this study, SIA emission was ignored, as the dissociation energy is high for low He/V ratios, which were most common [33].

## 4.4 He-V Cluster Mobility Models

Finally, bubble mobility is a third important area of significance that is debated in the literature [71]. Many studies have found that small SIA cluster mobility is significant and necessary to obtain correct results in damage recovery stages [26, 52, 59]. Other studies have found that small vacancy cluster mobility is not significant and does not play a significant role in defect evolution [51]. While still other studies have found that both large SIA and vacancy clusters should not be neglected in damage evolution simulations [72]. Similar to the other models, different models of cluster migration are considered in this study.

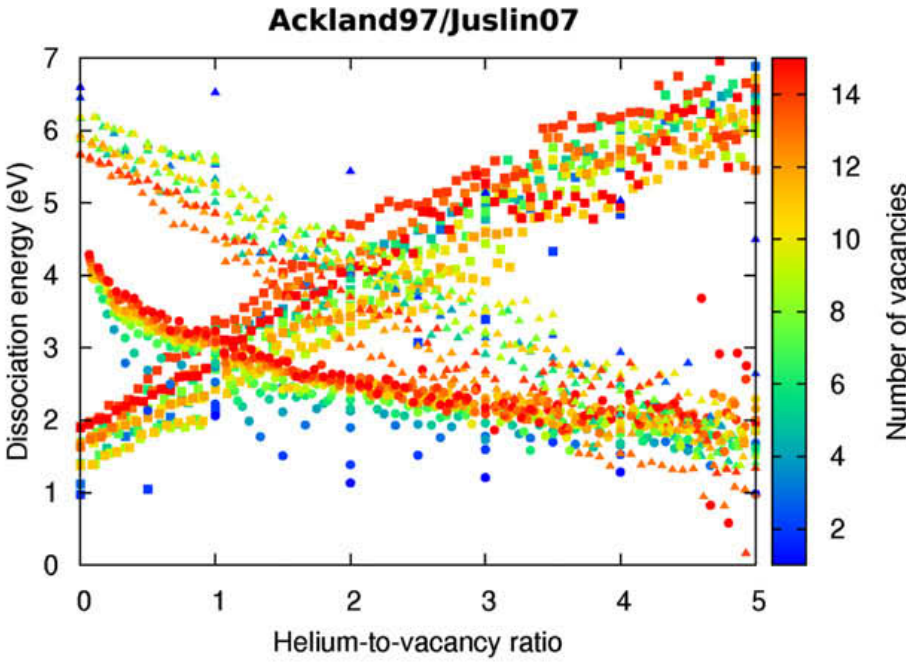



**Figure 4.1:** Dissociation energies calculated by MD simulation. Squares indicate vacancy dissociation, circles indicate helium interstitial dissociation, and triangles indicate SIA emission.

### 4.4.1 Full Mobility Model

The first model is based on the combined cluster mobility model from Domain [52], and behaves similar to the pure vacancy cluster mobility described in section 4.1.1. In this model, the migration energy $E_{m,He-V}$ of the $He_mV_n$ cluster is again kept constant and equal to the single vacancy migration energy $E_{m,V}$, while the attempt frequency decreases with helium population following a geometric progression of common ratio $q^{-1}$ to the corresponding vacancy cluster frequency $\nu_V(n)$:

$$r_{He_mV_n} = \nu_{He-V}(m,n)\exp\left(-\frac{E_{m,V}}{k_B T}\right) \tag{4.9}$$

$$\nu_{He-V}(m,n) = \nu_0\left(p^{-1}\right)^{n-2}\left(q^{-1}\right)^{m-1} \tag{4.10}$$

The progression factor $q$ was taken to be $q = 10$, which is based on KMC-calculated diffusivity values for small $He_mV_n$ clusters [73].

### 4.4.2 No He-V Mobility Model

The second model is based on the common assumption that He-V clusters are immobile [73]. In this model, $He_mV_n$ of all sizes are immobile, while all $V_n$ and $I_n$ clusters move as described previously.

### 4.4.3 Limited Cluster Mobility Model

The third model is the one used by Deo and Okuniewski [11, 50], and assumes that He does not affect cluster mobility. In this model, all $He_mV_n$ clusters behave identically to $V_n$ clusters as described in section 4.1.1. This model also assumes that SIA clusters are only mobile up to size $I_4$, and move with constant migration rate $E_{m,I_n} = 0.1\,\text{eV}$.

## 4.5 Annealing Results and Discussion

### 4.5.1 Vacancy-Only Radius Comparison

To compare the effect of the defect interaction radius on cluster formation, simulations were first performed considering only vacancy clustering (i.e., no cluster mobility or dissociation). The first two models used in this comparison were the constant radius and variable vacancy radius R(V) detailed in sections 4.2.1 and 4.2.2, respectively. The third model was a variation of the R(V) model which assumes that all defects have the same interaction radius corresponding to the R(V) $R_1$ radius, which was used to show the difference between the constant radius and variable radius models, and between two constant radius models with different radii.

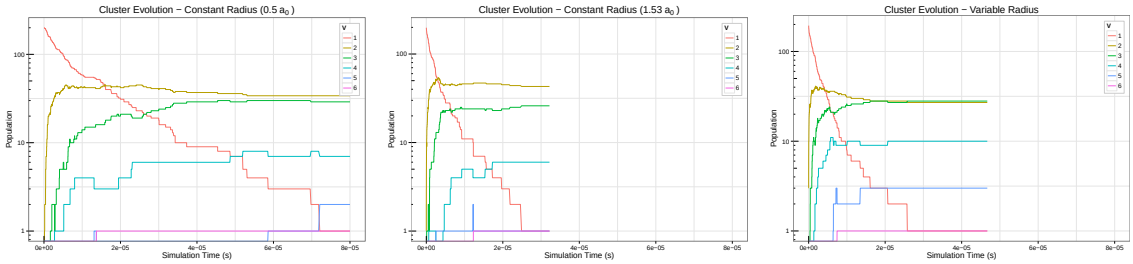
These simulations were performed in a smaller $100 \times 100 \times 100$ unit cell system at 543 K. Vacancies were added to the system in random initial positions to a concentration of 0.01 at%. Only single vacancies were assumed to be mobile in this study, with the vacancy migration energy taken from ab initio calculations to be 0.69 eV [74]. The simulations were run until all of the individual vacancies had become part of a cluster, which ranged from approximately 100,000 KMC steps for the fastest system (R(V)) and 500,000 KMC steps for the slowest system (constant radius).

The cluster population evolutions from the KMC simulations of the three interaction radius models are shown in fig. 4.2. The number of KMC steps required to complete the clustering differed considerably between the three simulations, however since the KMC algorithm is able to track a real time for the simulation, a comparison between the clusters population could be

made between the three models over the same times. The amount of real time required for all of the initial vacancies to cluster was different for each model, so to make a more meaningful comparison, the populations were compared up until the time when the fastest-clustering model (R(V)) finished.

Comparing the constant radius interaction models, differences in the cluster evolution were seen between the shorter range ($r_n = 0.5\,a_0$) constant radius model and the longer range ($r_n = 1.53\,a_0$) modified R(V) model. The biggest difference was that in the time scale compared, the constant radius model did not fully cluster. Approximately 15 % of the initial vacancies were still present as single vacancies in the constant radius model when all of the vacancies in the modified R(V) model had clustered. The rate at which the single vacancies combined into clusters was also considerably faster in the modified R(V) model. More vacancies clustered within the first 25 % of the simulation in the modified R(V) model than would ever cluster in the constant radius model. The $v_3$ clusters also appeared earlier and their population grew faster in the modified R(V) model compared to the constant radius model. From these trends it was concluded that the larger constant interaction radius increases the cluster formation rate for small clusters ($v_2$,$v_3$), but does not help the growth of larger clusters.

Comparing the constant-radius modified R(V) model with the original variable-radius R(V) model, differences were seen in the evolution of larger clusters compared to smaller clusters. The initial formation of smaller clusters ($v_2$,$v_3$) was similar in the variable radius model compared to the modified R(V) model, however in the R(V) model, larger clusters appeared earlier in the simulation. The growth of larger clusters was also much stronger in the R(V) model, with larger clusters appearing earlier, growing faster, and ending in much larger populations compared to the modified R(V) model. From these trends it is concluded that the R(V) model increases the initial formation and growth rate of larger clusters and results in a higher concentration of larger clusters compared to the modified R(V) model.

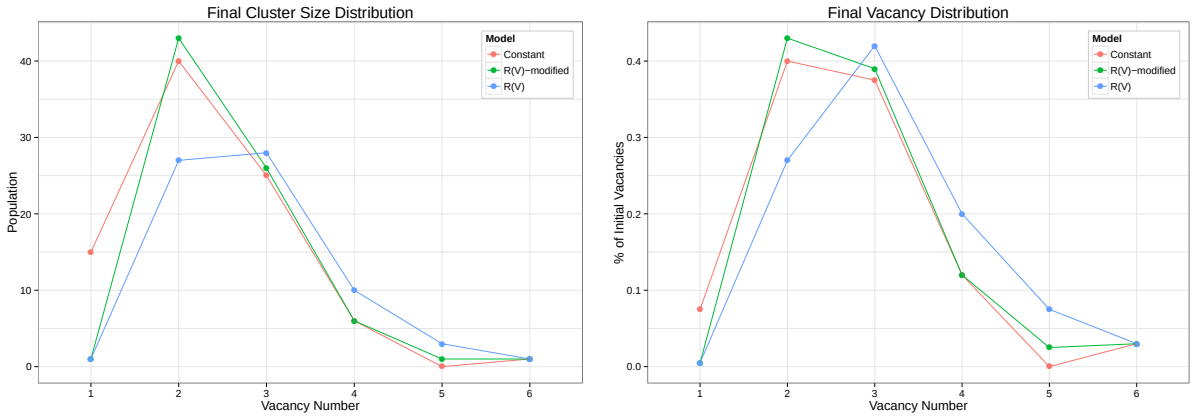



**(a)** constant radius **(b)** R(V) (modified) **(c)** R(V)

**Figure 4.2:** KMC simulation results for the (a) constant radius model, (b) constant-radius modified R(V) model, and (c) R(V) model. Cluster size populations tracked over the life of the simulation. Legend entries indicate the number of vacancies in the cluster.

To compare the final state of the three interaction radius models, the final cluster size distribution and the final vacancy distribution are shown for the three interaction models (again at the fastest simulation end time) in figs. 4.3a and 4.3b, respectively. These results show that the final cluster size distribution is shifted toward fewer but larger clusters with increasing cluster radius, especially in the variable radius case. This shift is in agreement with the results by Okuniewski, who found a similar low-density large-size cluster distribution experimentally.

The clusters are still smaller in all three KMC simulations than what was found experimentally, but this is very clearly due to differences in the simulation conditions. The experimental results are based on irradiated samples, while these simulations were in a smaller system and only contained annealing without the irradiation and the continuous introduction of new damage.




**(a)** Cluster Distribution **(b)** Vacancy Distribution

**Figure 4.3:** Final population distribution for the three interaction radius models showing (a) the cluster size distribution and (b) the vacancy distribution.

From the cluster evolution trends it is concluded that between constant interaction radius models, the larger constant radius model increases the cluster formation rate for small clusters ($v_2$,$v_3$), but does not help the growth of larger clusters. It is also concluded that between the constant and variable interaction radius models, the variable radius model increases the initial formation and growth rate of larger clusters and results in a higher concentration of larger clusters compared to the constant radius model. From the shift toward low-density large-size final cluster size distributions, it is concluded that the R(V) better reproduces the cluster distribution trends found experimentally.

### 4.5.2 Stages of Equilibration

All subsequent results are based on simulations performed using the conditions described in section 4.1.4. Before exploring the parameters to be considered in this study, the various phases of the equilibration simulations are first considered. As an example, consider the time series evolution of an equilibration run using the R(V) radius model, variable dissociation model, and full cluster mobility model, with 10 %He. This system was evolved for $8 \times 10^6$ KMC steps, with the results shown in figs. 4.4 and 4.5.

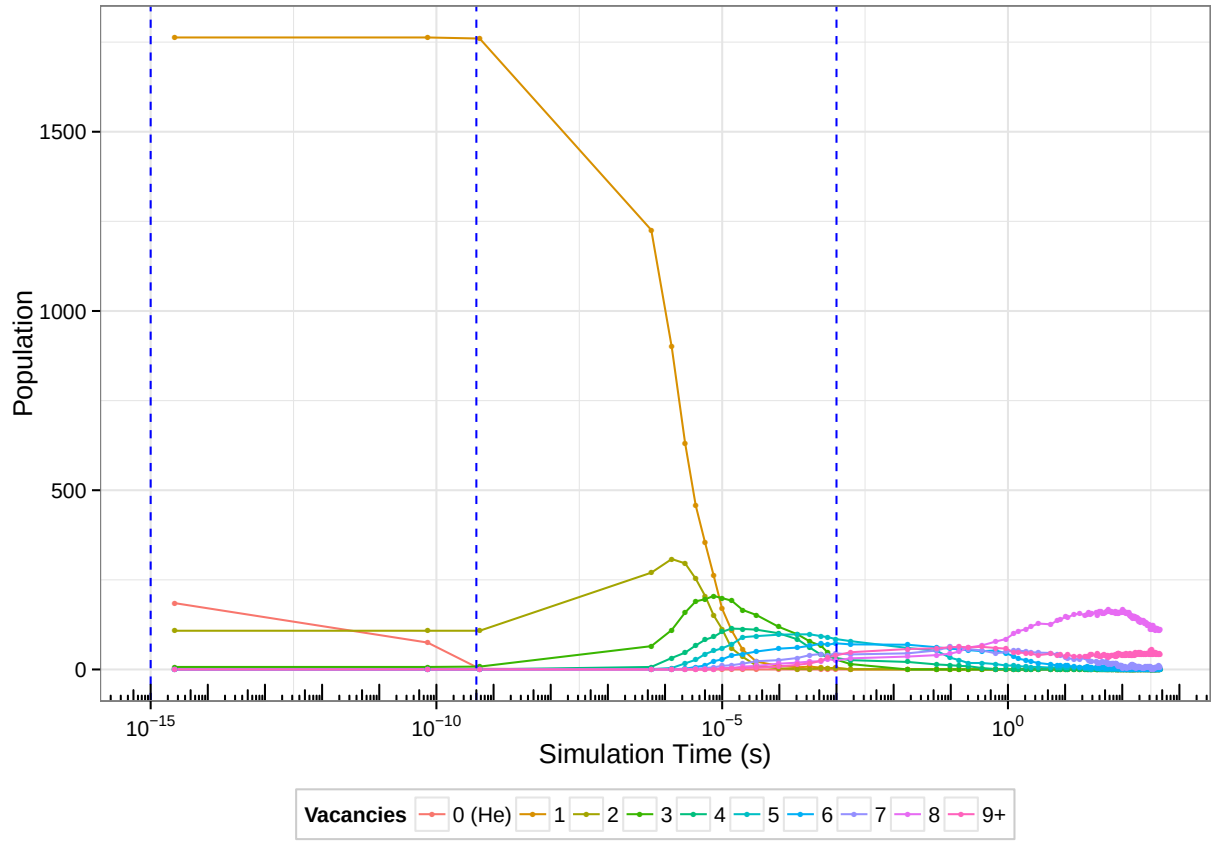
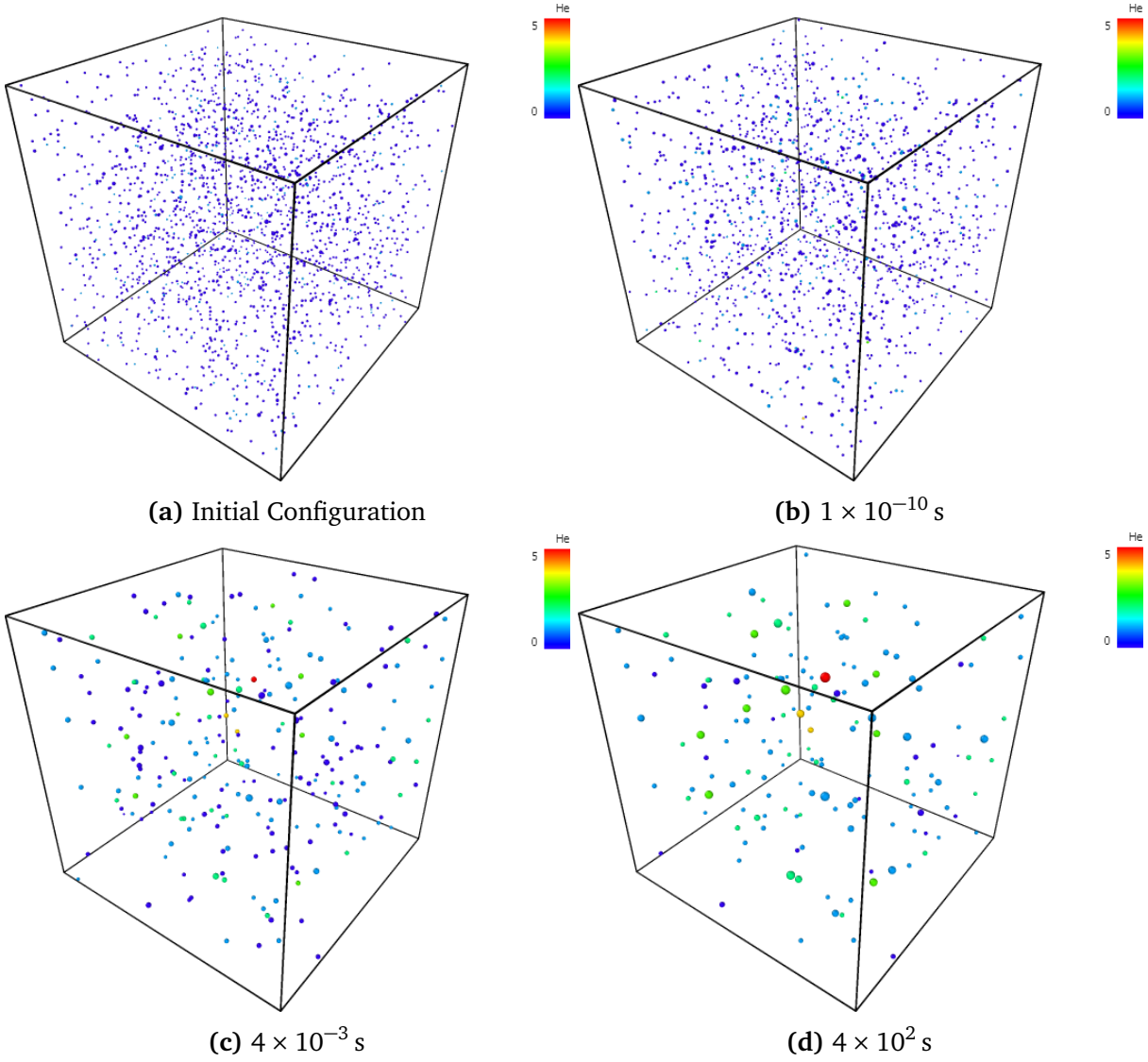



**Figure 4.4:** Time series results that demonstrate stages of system evolution. All $He_mV_n$ clusters with the same number of vacancies are considered together. Stages of evolution are separated by dashed blue lines.

The evolution of the system for this evolution goes through three distinct major stages of evolution. The system is initialized with 2000 vacancies and 200 helium interstitials, all randomly distributed throughout the system. As the migration energy of helium is significantly smaller than the migration energy of iron vacancies, the first stage of evolution is comprised primarily of helium migration. The helium migrates until it interacts with a vacancy or vacancy




**(a)** Initial Configuration

**(b)** $1 \times 10^{-10}$ s

**(c)** $4 \times 10^{-3}$ s

**(d)** $4 \times 10^{2}$ s

**Figure 4.5:** He-V defect cluster configurations at (a) 0 s (Initial Configuration), (b) $1 \times 10^{-10}$ s (Stage 2), (c) $4 \times 10^{-3}$ s (Stage 3), and (d) $4 \times 10^{2}$ s (Final). Cluster size is based on the number of vacancies in the cluster, and color is based on the number of helium in the cluster.

cluster, forming a $He_mV_n$ cluster. This is clearly seen in the example evolution, as only the helium population ($V = 0$) changes, decreasing monotonically during the first stage. This process continues until most of the helium has interacted and formed some cluster, at which point the system transitions to the second stage.

In the second stage of evolution, iron vacancies begin to become mobile. Like the first stage, vacancies migrate until they have mostly interacted and formed either $He_mV_n$ or $V_n$ clusters. This is clearly seen in the example evolution, as the single vacancy population decreases monotonically, giving rise to small ($V \lesssim 6$) $He_mV_n$ clusters. These clusters are also seen to grow

as the single vacancies interact with them and grow.

In the third stage, the cluster mobility and dissociation start to become favorable. For clarity, all clusters of size $V \geq 9$ are represented as a single line in the example plot, however the clusters in this evolution formed up to $V = 43$. Clusters remain at relatively stables levels briefly as the clusters start to migrate and single vacancies dissociate from clusters. This continues until clusters start to interact with each other, at which point the clusters begin to grow and form large clusters. This phase is the most variable over the various parameter sets, and will be explored in more detail in the following sections.

There is one aspect of this evolution that should be clarified at this point, as its influence will appear in later results. As seen in fig. 4.4, when the variable dissociation energy model is used, a concentration of $V_8$ peaks near the end of the simulation time. The concentration peak leads to unusually shaped size distributions, but there are some things that help explain its presence. First, the building up of $V_8$ clusters seems to form initially because of the high dissociation energy of $V_8$ clusters in the variable dissociation energy model. The full set of dissociation energies is shown in fig. 4.1, with the vacancy dissociation energy for the lowest He ratio (most common), shown in fig. 4.6. In this energy set, $V_8$ clusters have the highest dissociation energy compared to its surroundings, and thus as the clusters continue to grow, $V_8$ can be expected to accumulate. Second, after the $V_8$ accumulates and the concentration peak forms, it does start to dissipate. This indicates that given additional annealing time, the peak would reduce to a less extreme level relative to the concentrations of neighboring cluster sizes. Since the goal of this study was to compare the effects of different parameter models between each other, it was decided that computational time was better spent comparing different models instead of increasing the run times to shrink the $V_8$ concentration peak.

### 4.5.3 Radius Model Comparison

From the vacancy-only simulations in section 4.5.1, it has already been seen that the extended constant radius model did not help the constant radius model form larger clusters. In this section, the original constant radius model and three variable radius models discussed in section 4.2 are compared with each other and with different concentrations of helium. In each case, the variable dissociation energy model and full cluster mobility model were used.

Results are first compared for a specific helium concentration with varying cluster model. Figure 4.7 shows the time series evolution of the four different radius models for a helium to vacancy ratio of 1:10. Similar evolution trends were observed in each model, and each model followed the general stages outlined in the previous section. However, just as with the

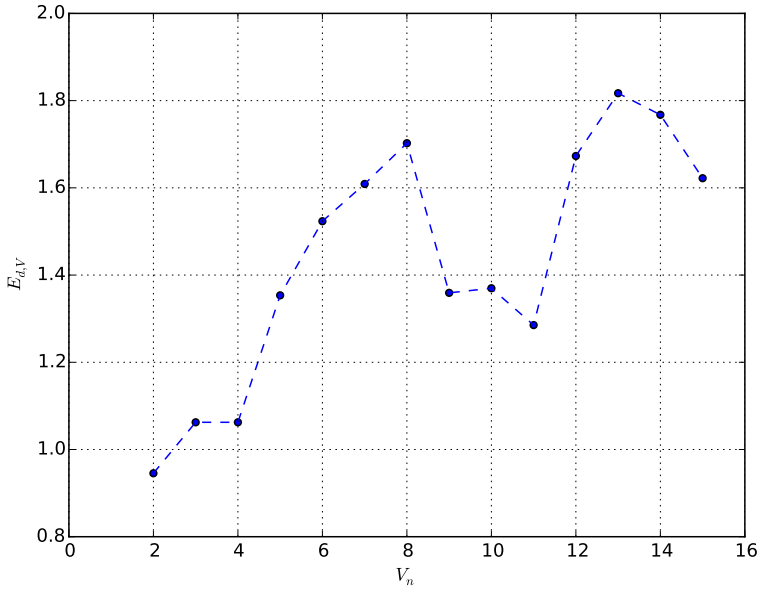



**Figure 4.6:** Vacancy dissociation energy for $V_nHe_0$. $V_8$ appears as a local maximum relative to its surrounding.

vacancy-only simulations, the constant radius model differed significantly from the variable radius models. While the constant radius model evolved similarly to the variable radius model, it hit the key evolution points around an order of magnitude later in each case. The constant radius model also took significantly more KMC steps to evolve to each realtime point. All model simulations were initially run for $4 \times 10^6$ KMC steps, which is approximately as many steps that could be completed within the runtime limit of the cluster used for the simulations. While the variable radius models all made it to about the same point in their evolution, the constant radius model only evolved up to the first dashed red line in the figure. The variable radius models where extended for an additional $4 \times 10^6$ KMC steps in order to extend the evolution past the building up of $V_8$ clusters. The constant radius model simulation was extended an additional $4 \times 10^6$ KMC steps several times in order to reach the same realtime level for comparison with the variable radius models. However after $2 \times 10^7$ KMC steps (5x simulation time) the constant radius model had still not reached the first stop point of the variable radius models. These resume points are all shown visually in fig. 4.7 with red dashed lines. This further emphasizes the importance of the radius model. In the constant radius model, particles spend a large number of KMC steps migrating around each other without interacting, resulting in significantly longer run times.

The difference in the clustering behavior is also apparent when comparing the cluster size distributions at different times. This comparison is done using the 1:10 He:V ratio results as this

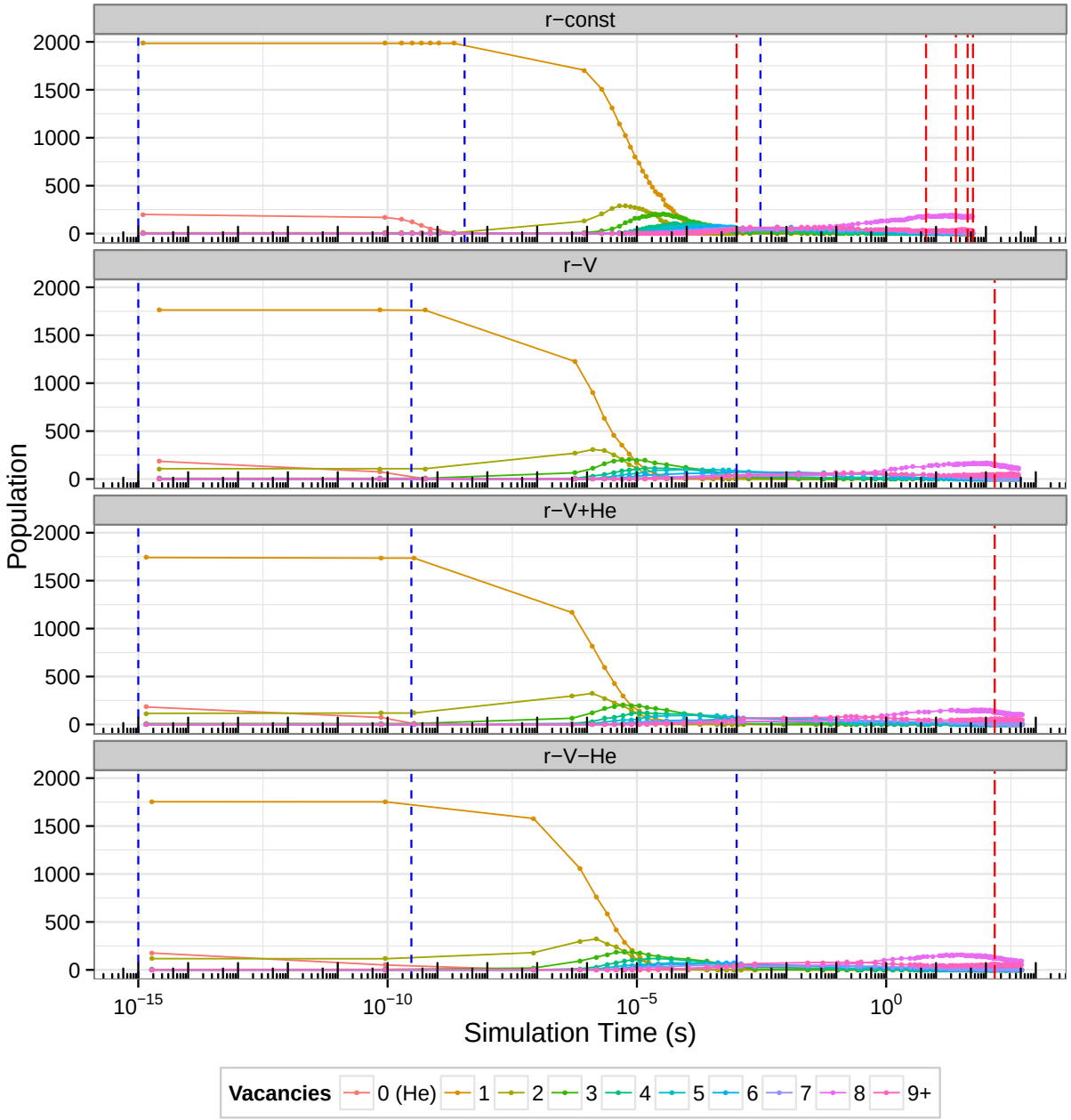



**Figure 4.7:** Cluster radius model time series comparison. Clusters are aggregated based on number of vacancies. Blue dashed lines indicate evolution stages. Red dashed lines indicate simulation resume points.

was the highest concentration of helium studied and should show the greatest impact of helium in the helium-dependent models. in the initial configuration ($t = 0\,\mathrm{s}$, fig. 4.8), all simulations start with the same number of vacancies and helium randomly distributed. The variable radius models started with clusters after the random distribution, while the constant radius model started with only a few $He_mV_2$ clusters. At the midpoint of the simulation ($t \approx 1 \times 10^{-5}\,\mathrm{s}$, fig. 4.9), the lagging trend became clear. At the same time point, the constant radius model consisted mostly of single or small vacancy clusters, while the variable radius models show a transition to larger clusters. At the final time comparison ($t \approx 31\,\mathrm{s}$, fig. 4.10) the constant radius model had started to catch up with the trends of the variable radius models, but was still shifted towards smaller clusters.

Looking at the variable radius models, the cluster size distributions overlapped significantly. In the initial and midway comparisons (figs. 4.8 and 4.9) the distributions were almost identical. Looking at the final distribution (fig. 4.10), there was a slight shift upward toward larger clusters using the R(V+He) model compared to the others, but for the most part, the R(V+He) and R(V-He) model distributions are within error bars of the R(V) model. From these trends it is concluded that the variable radius models produce size distributions which are more consistent with the large size low density distributions found experimentally. It is also concluded that the constant radius model lagged behind the variable radius models in each case, and still resulted in more small clusters and fewer large clusters.

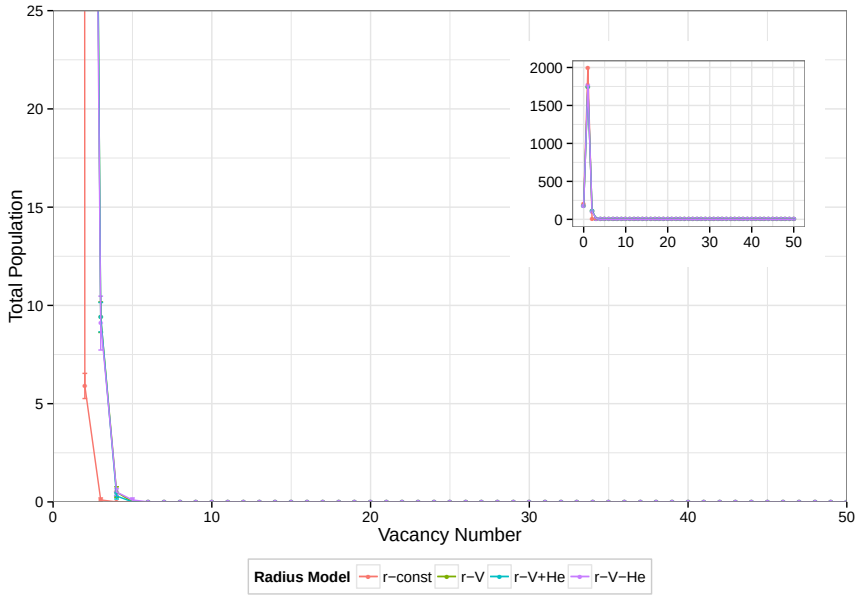



**Figure 4.8:** Initial cluster size distribution for varying radius models for 1:10 He:V ratio. Clusters are aggregated based on number of vacancies.

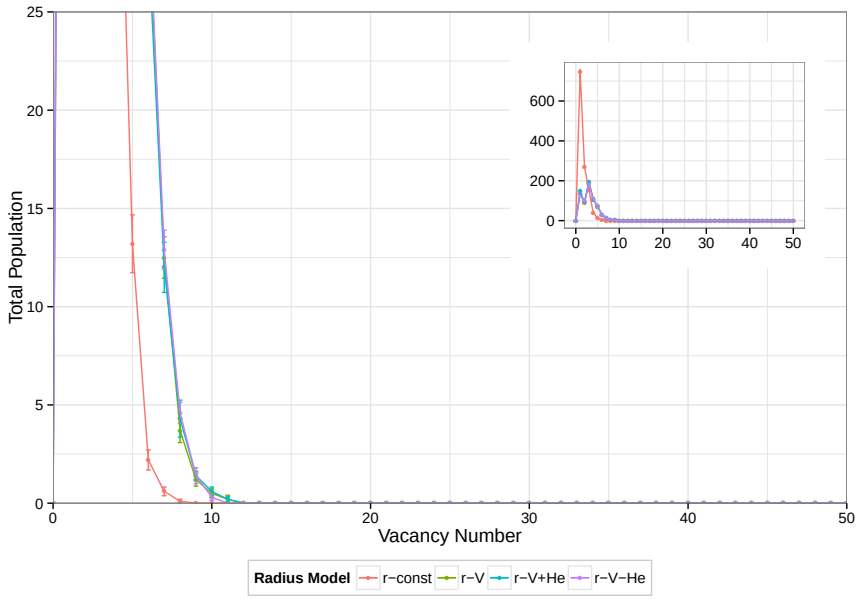



**Figure 4.9:** Midway ($t \approx 1 \times 10^{-5}$ s) cluster size distribution for varying radius models for 1:10 He:V ratio. Clusters are aggregated based on number of vacancies.

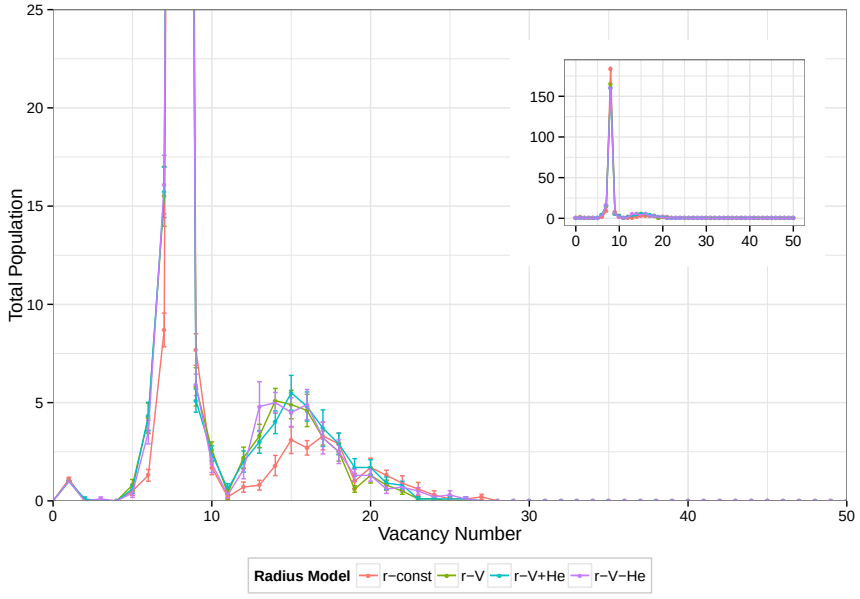



**Figure 4.10:** Final ($t \approx 31$ s) cluster size distribution for varying radius models for 1:10 He:V ratio. Clusters are aggregated based on number of vacancies.

As mentioned earlier, the 1:10 He:V ratio results were used in this comparison as if there were variations seen between the helium-dependent model and the vacancy only model then these variations would be most prominent in the highest helium concentration results. Even in this case, the R(V+He) and R(V-He) radius distributions overlapped the R(V) distribution at each stage. This is likely due to the large difference in atomic volume between iron and helium (5.65 $Å^3$ vs. 0.49 $Å^3$). Both of the helium-dependent models are based on an effective cluster volume that is either the sum or difference of the volume of cluster constituents, which would require approximately 10 helium atoms to have the same effect on radius as a single iron atom. Thus it is concluded that for the helium concentrations in this study, the addition of helium dependence in the variable radius models did not significantly affect the distribution when compared with the non-helium dependent model, even with the highest concentration of helium present.

### 4.5.4 Dissociation Model Comparison

In this section, the constant dissociation model and variable dissociation model described in section 4.3 are compared. Based on the radius model results from section 4.5.3, these simulations were performed using the R(V) radius model. These simulations were also performed based on the full cluster mobility model described in section 4.4.1. This was done to limit the effect of the mobility restrictions imposed from the other models when considering dissociation effects. The cluster size distributions for different dissociation energy models are compared first for a specific helium concentration in this section, then between helium concentrations section 4.5.6.

Figures 4.11 and 4.12 show the midpoint ($t \approx 1 \times 10^{-5}$ s) and final ($t \approx 320$ s) cluster size distributions for the 1:10 He:V ratio and the full cluster mobility model. The two dissociation energy models overlapped almost completely at the midpoint, but resulted in clearly different final distributions. The constant-energy model resulted in a relatively narrow peak with a higher density of smaller clusters, while the variable-energy model resulted in relatively wide distribution with a lower density of larger clusters. The variable-energy model also resulted in a sharp peak at $V_8$ which was not observed in the constant-energy distribution. This is likely due to the relatively high vacancy dissociation energy for $V_8$ clusters compared to neighboring sizes as described earlier, and will be discussed further shortly.

To better illustrate the difference in evolution, fig. 4.13 shows the time series evolution of the two dissociation energy models for the 1:10 He:V ratio and full mobility model. Both parameter sets were initially run for the same number of KMC steps ($4 \times 10^6$ steps), which yielded significantly different results. The variable radius model was run for an addition $8 \times 10^6$

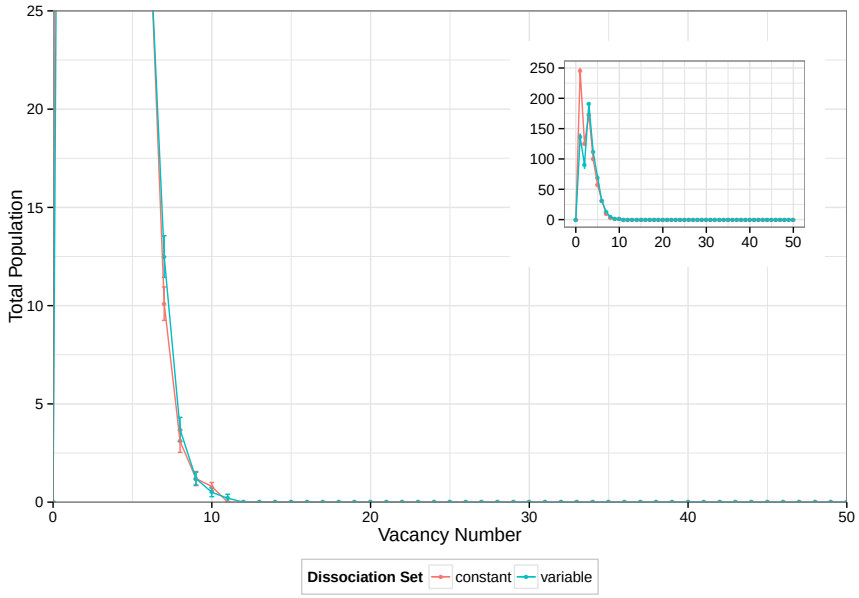



**Figure 4.11:** Midpoint ($t \approx 1 \times 10^{-5}$ s) cluster size distribution for varying dissociation energy models for 1:10 He:V ratio and full cluster mobility model. Clusters are aggregated based on number of vacancies.

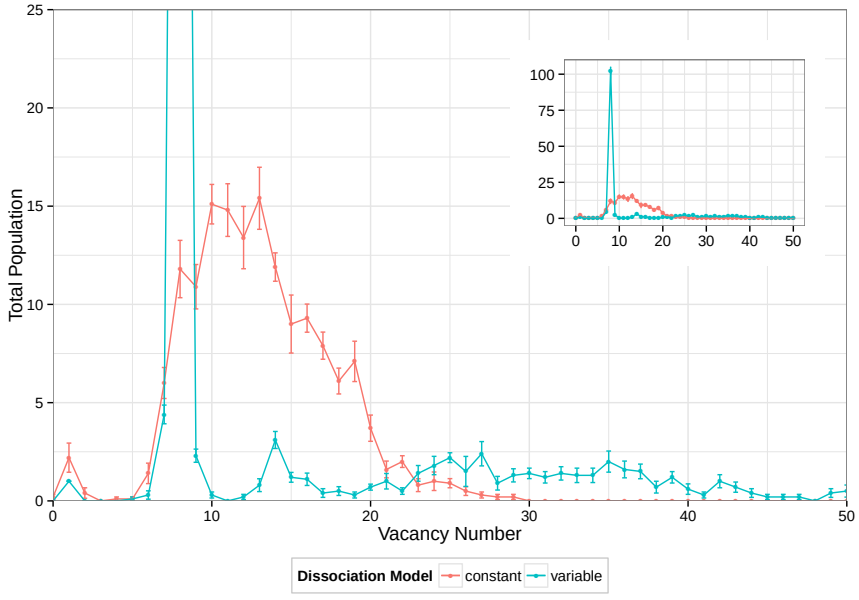



**Figure 4.12:** Final ($t \approx 600$ s) cluster size distribution for varying dissociation energy models for 1:10 He:V ratio and full cluster mobility model. Clusters are aggregated based on number of vacancies.

steps in an attempt to reach the same realtime. While the result was clearly still significantly behind the constant-energy results, the constant-energy results had remained relatively stable for the remaining real time, so the distribution comparison in fig. 4.10 was done at the final time for the variable-energy results.

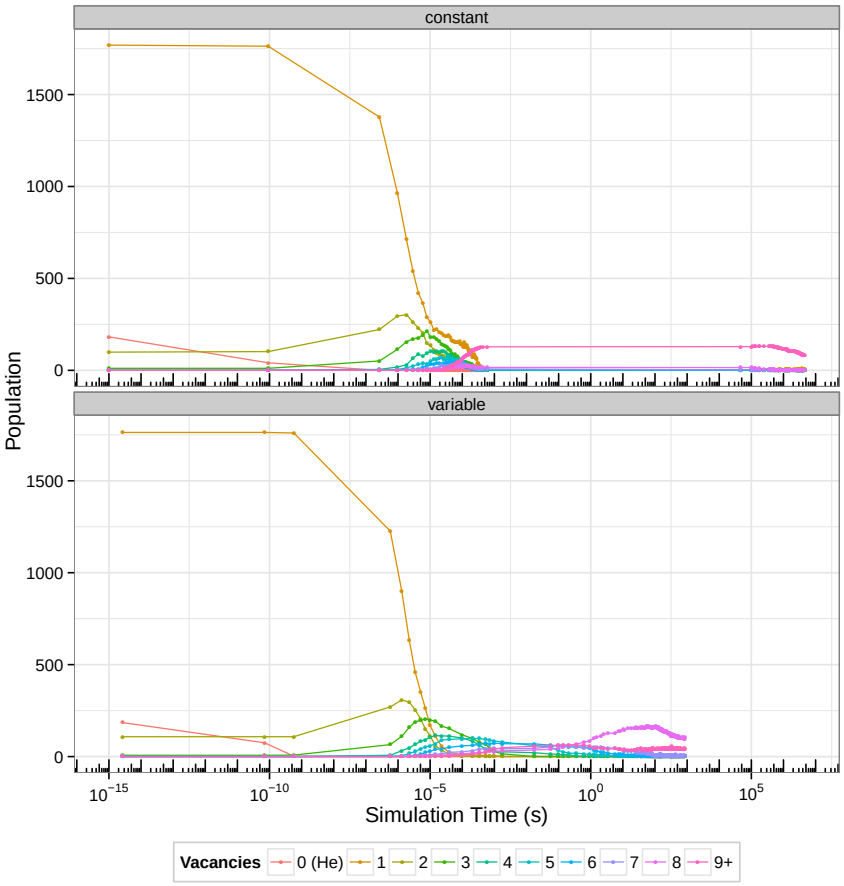



**Figure 4.13:** Dissociation energy model time series comparison for 1:10 He:V ratio and full mobility model. Clusters are aggregated based on number of vacancies.

Looking further at the time series results, both dissociation energy models showed similar trends at short times ($t \lesssim 1 \times 10^{-5}$ s). Around this time, the initial vacancy clustering is finishing up and cluster mobility/dissociation starts to become important, at which point the trends quickly diverge. In the constant-energy model, all clusters form in a relatively short time frame, then remain constant for a relatively long time scale, then finally start to evolve again. In the variable-energy model, small clusters form initially, then begin shifting from small clusters to larger clusters. Small clusters decrease as $V_8$ clusters increase sharply, in addition to growth of some larger clusters. Finally $V_8$ clusters start to decrease as larger clusters increase.

These different trends can be explained by considering the dissociation energies compared

to the cluster mobility. In the constant-energy model, the vacancy dissociation energy from $V_n$ clusters is small (0.89 eV) and similar to the vacancy migration energy (0.69 eV), while the vacancy dissociation energy from $He_mV_n$ clusters is large (2.69 eV) compared to vacancy migration. As the constant-energy simulation begins, small $V_n$ and $He_mV_n$ clusters form and can migrate, but only the $V_n$ clusters are able to dissociate at that time scale. The $V_n$ clusters migrate and dissociate, eventually coalescing with the $He_mV_n$ which are unable to dissociate. When coalescence results in no further possible dissociation or migration at the small time scale, there is a large jump in time scale until $He_mV_n$ dissociation and large cluster migration become possible, at which point evolution continues along these mechanisms.

In the variable-energy model, the vacancy dissociation energies from $He_mV_n$ clusters (including $V_n$) are more closely spaced, and for lower He concentrations are around 1.0 eV to 1.8 eV. The migration energy is highest for $V_8$ clusters as was discussed in section 4.5.2, which is likely the cause of the observed spike in concentration. As with the constant-energy model, small clusters form and begin to migrate and dissociate on about the same time scale. They begin to coalesce and form primarily $V_8$ clusters, along with some larger clusters. Then as the time scale reaches the $V_8$ dissociation energy the $V_8$ clusters begin to dissociate or grow to sufficient sizes that they continue to grow.

### 4.5.5 Cluster Mobility Comparison

In this section, the three cluster mobility models described in section 4.4 are compared. Based on the radius model results from section 4.5.3, these simulations were performed using the R(V) radius model. These simulations were also performed using both of the dissociation energy models in order to verify any dependence on these models. As with the dissociation energy model comparison, the cluster mobility models are compared first for a specific helium concentration in this section, then between helium concentrations in section 4.5.6.

Figure 4.14 shows the time series evolution for the 1:10 He:V ratio and the variable dissociation energy model. Unlike in earlier comparisons, the three cluster mobility models compared followed very similar evolution phases for most of the simulation, only diverging significantly near the end point.

Figures 4.15 and 4.16 show the midpoint ($t \approx 1 \times 10^{-5}$ s) and final ($t \approx 350$ s) cluster size distributions for the 1:10 He:V ratio and the variable dissociation energy model. The three cluster mobility models overlapped almost completely at the midpoint, but resulted in varying final distributions. All three distributions displayed the same $V_8$ peaking behavior as a result of the dissociation model used, but differed in the larger cluster distribution. The full and limited

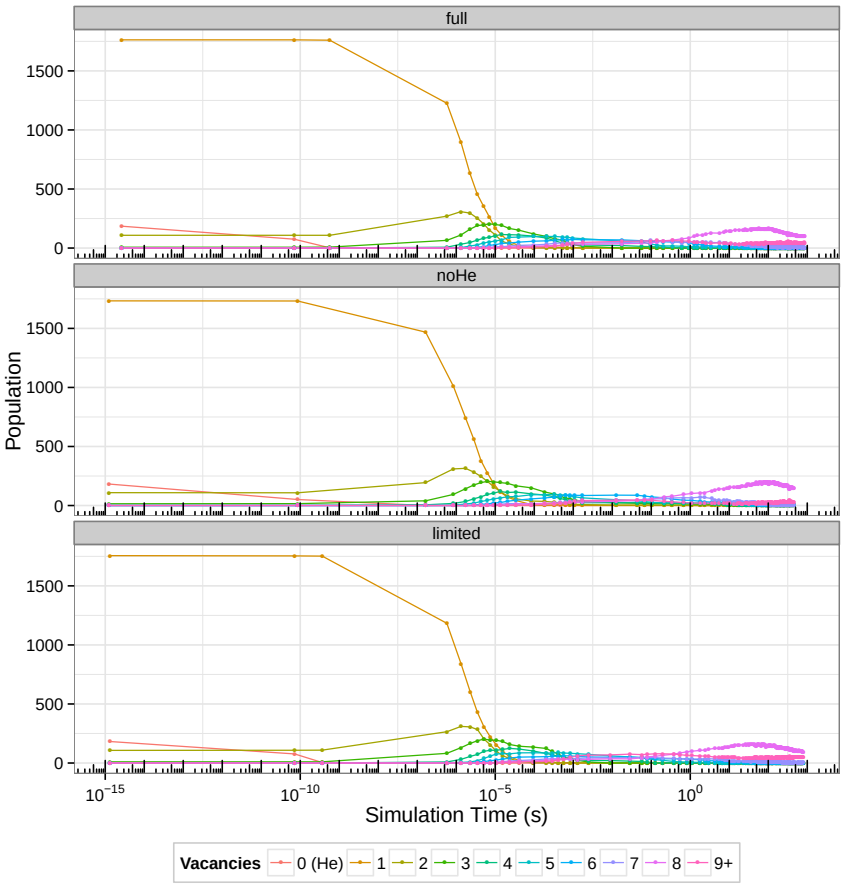



**Figure 4.14:** Cluster mobility model time series comparison for 1:10 He:V ratio and variable dissociation energy model. Clusters are aggregated based on number of vacancies.

cluster mobility models mostly overlapped, while the immobile $He_mV_n$ cluster model resulted in lower density of smaller clusters by comparison.

Figure 4.17 shows the time series evolution for the 1:10 He:V ratio and the constant dissociation energy model. As was observed in the dissociation energy comparison, the constant dissociation energy model resulted in fast $He_mV_n$ clustering following by significantly longer real time evolutions as a result of lack of dissociation pathways.

Figures 4.18 and 4.19 show the midpoint ($t \approx 1 \times 10^{-5}$ s) and final ($t \approx 5 \times 10^{5}$ s) cluster size distributions for the 1:10 He:V ratio and the constant dissociation energy model. As with the variable dissociation energy comparison, the three cluster mobility models overlapped almost completely at the midpoint, but resulted in varying final distributions. Again the full and limited cluster mobility models mostly overlapped, while the immobile He-V cluster model resulted in a similar distribution shifted toward lower cluster sizes. As with the variable dissociation energy results, the overlap of the full and limited cluster mobility models is likely due to the

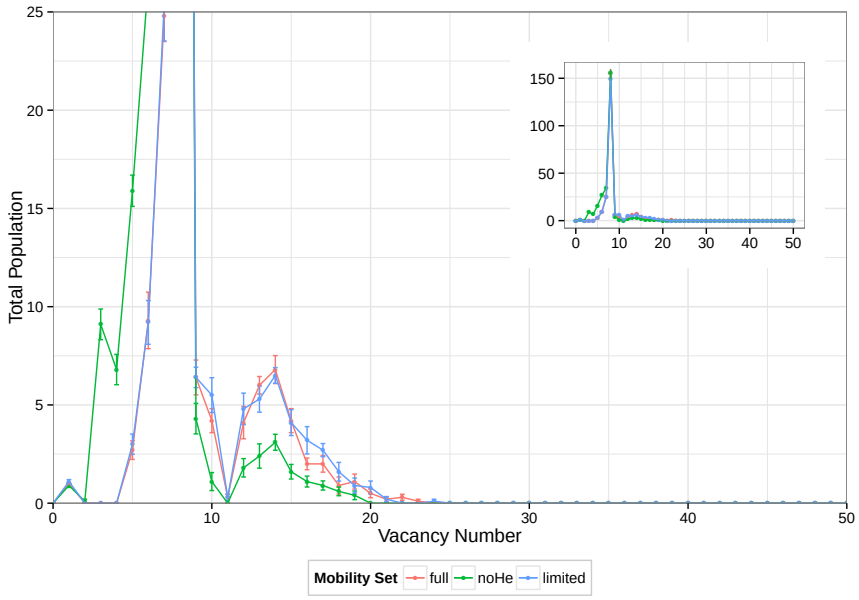



**Figure 4.15:** Midpoint ($t \approx 10\,\mathrm{s}$) cluster size distribution for varying mobility models for 1:10 He:V ratio and variable dissociation energy model. Clusters are aggregated based on number of vacancies.

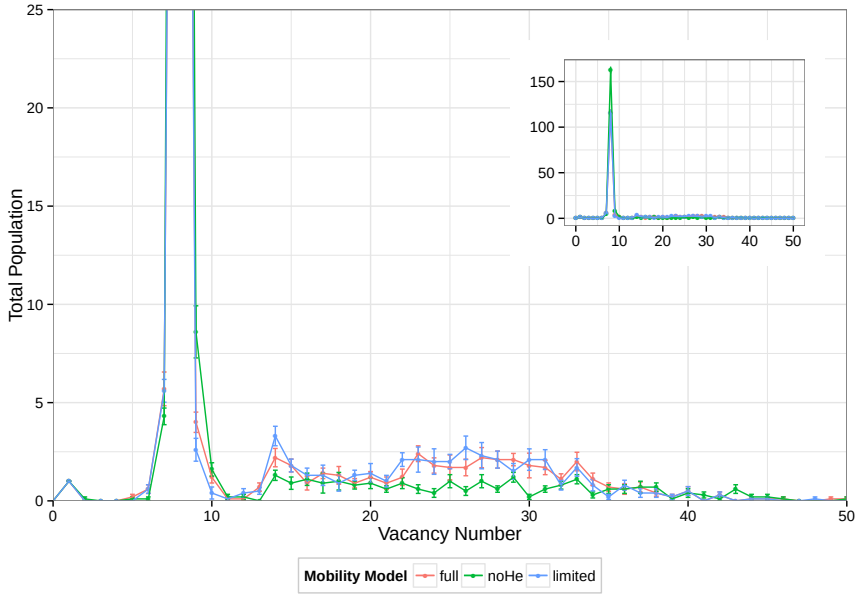



**Figure 4.16:** Final ($t \approx 350\,\mathrm{s}$) cluster size distribution for varying mobility models for 1:10 He:V ratio and variable dissociation energy model. Clusters are aggregated based on number of vacancies.

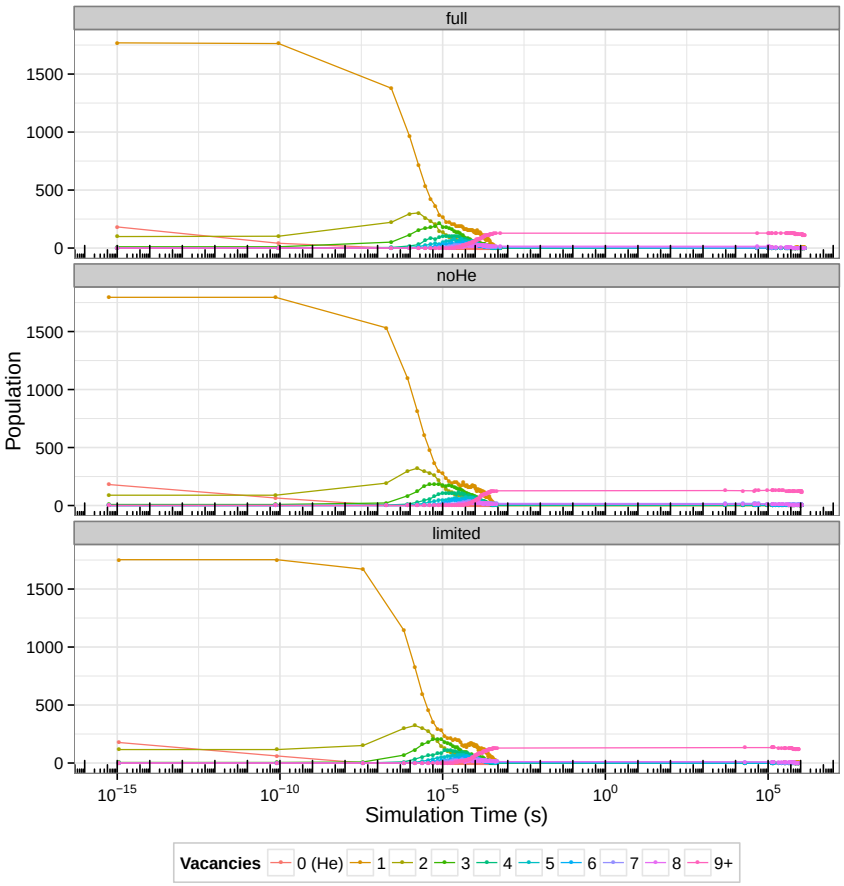



**Figure 4.17:** Cluster mobility model time series comparison for 1:10 He:V ratio and constant dissociation energy model. Clusters are aggregated based on number of vacancies.

similarity in the two models. The two significant differences in the limited model compared to the full model are that SIA clusters are only mobile up to size 4, and that $He_mV_n$ clusters behave as $V_n$ clusters. Since these annealing simulations are performed without the interstitials present, the difference in SIA cluster mobility has no affect. The difference that could appear at this stage is ignoring the effect of helium on the $He_mV_n$ clusters. However this change resulted in no significant change in the size distribution. Since the presence of helium acts to slow down cluster migration as helium is added, this result also shows that the helium content of the clusters remained relatively low.

### 4.5.6 Helium Concentration Comparison

Finally, the effect of helium concentration is compared for each of the migration energy model and cluster mobility model combinations. The distributions of the three helium concentrations

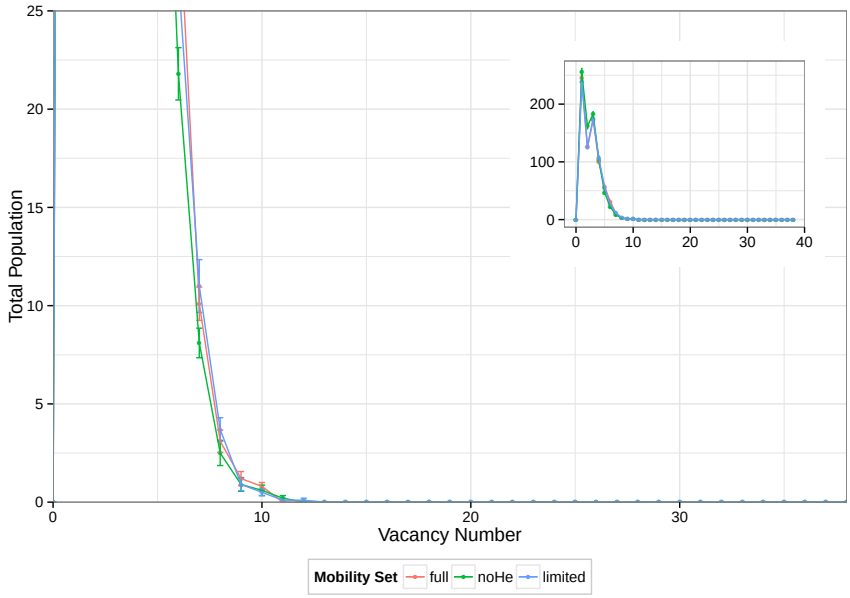



**Figure 4.18:** Midpoint ($t \approx 1 \times 10^{-5}$ s) cluster size distribution for varying mobility models for 1:10 He:V ratio and constant dissociation energy model. Clusters are aggregated based on number of vacancies.

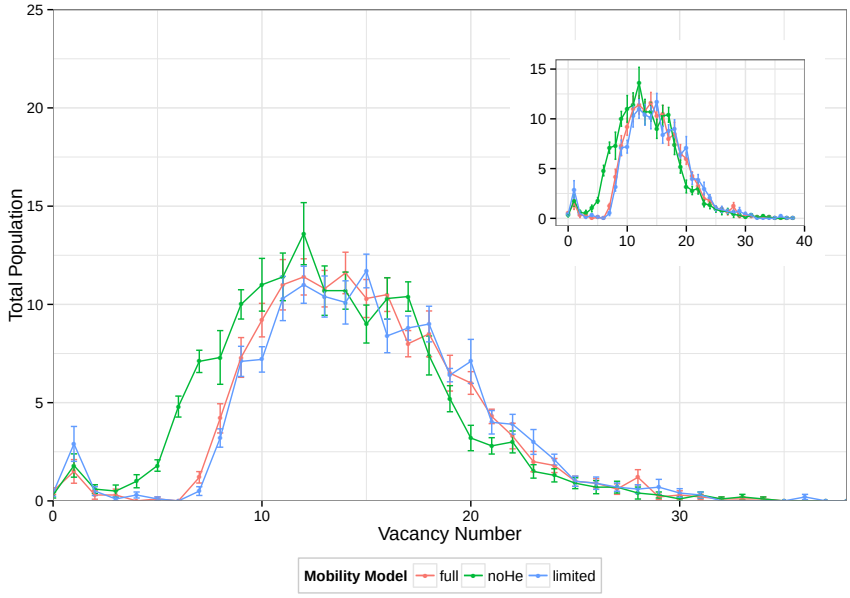



**Figure 4.19:** Final ($t \approx 5 \times 10^{5}$ s) cluster size distribution for varying mobility models for 1:10 He:V ratio and constant dissociation energy model. Clusters are aggregated based on number of vacancies.

(He:V ratios of 1:100, 1:20, and 1:10) are compared for each set of parameters. As was seen in earlier sections, the different parameters resulted in significantly different real time evolutions, so while the helium concentrations of a single parameter set are compared at the same real time, it was not meaningful to compare results from all parameter sets at the same real time.

The three cluster mobility models were compared first for the variable migration energy model. Figures 4.20, 4.21 and 4.22 show the final distributions for the full mobility model, limited mobility model, and noHe mobility model, respectively. The variable dissociation energy models all followed the same general form seen in fig. 4.16. Consistent with the mobility comparison for the variable migration energy model, the full mobility model and limited mobility model result in similar trends. The strongest shift was observed in the noHe mobility model. Since $He_mV_n$ clusters are assumed to be immobile in this model, the increased helium concentration combined with the strong mobility trapping creates more small trapped clusters.

When comparing distributions of differing He:V ratios, the distributions all followed the same general trend: increasing the helium concentration shifts the size distribution down from smaller numbers of larger clusters to larger numbers of small clusters. This is likely due to the trapping effect of the helium. Vacancies bind to the helium which makes it more difficult to dissociate or migration to find larger clusters. Rather than coalescing, these defect act as traps for mobile defects, resulting in a larger number of smaller clusters. This is consistent with the trend seen experimentally by Okuniewski [11] in fig. 4.23. These results are based on much higher defect concentrations (after 3 dpa) than used in these annealing runs, however qualitatively the shift towards a higher density of smaller clusters is consistent.

The three cluster mobility models were then compared for the constant migration energy model. Figures 4.24, 4.25 and 4.26 show the final distributions for the full mobility model, limited mobility model, and noHe mobility model, respectively. The constant dissociation energy models all followed the same general form seen in fig. 4.19. Unlike in the variable migration energy results, the distinction between the full mobility model and limited mobility model results compared with the noHe mobility model was much less significant for larger clusters in the constant migration energy results.

When comparing distributions of differing He:V ratios, the distributions all followed the same general trends, but these trend was different from the variable migration energy model and the experimental results. The first immediate difference is the behavior at the lowest helium concentration. While the 1:10 and 1:20 He:V ratio distributions formed broad peaks with a small speak at the lower end corresponding to mobile defects, the 1:100 He:V ratio distribution formed a sharp peak at the lower end with the population consisting primarily of small clusters. Initially, there was also a sharp peak at the upper end at $He_mV_{50}$ (see fig. 4.27), which was believed to be likely due to the limitation on the maximum size of clusters considered in the

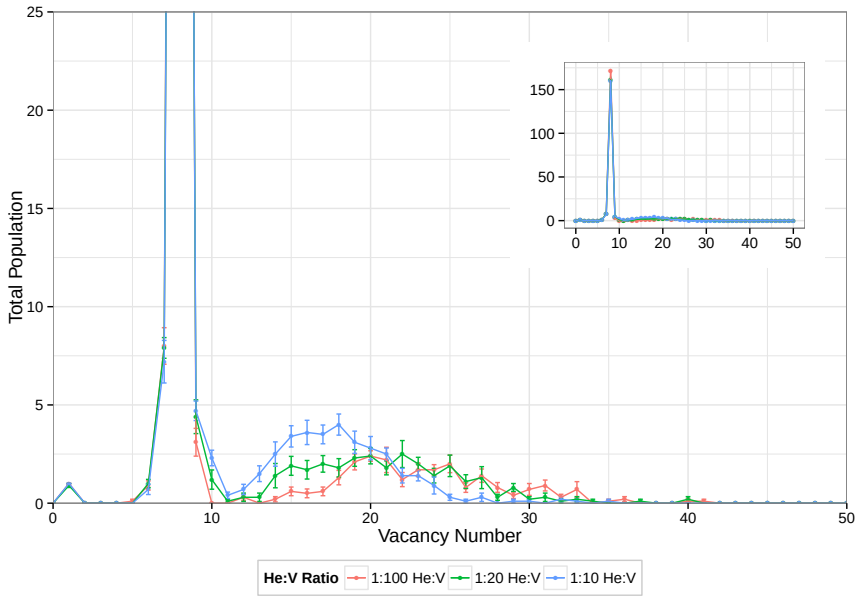



**Figure 4.20:** Final ($t \approx 90\,\mathrm{s}$) cluster size distribution for varying He:V ratios for variable dissociation energy model and full cluster mobility. Clusters are aggregated based on number of vacancies.

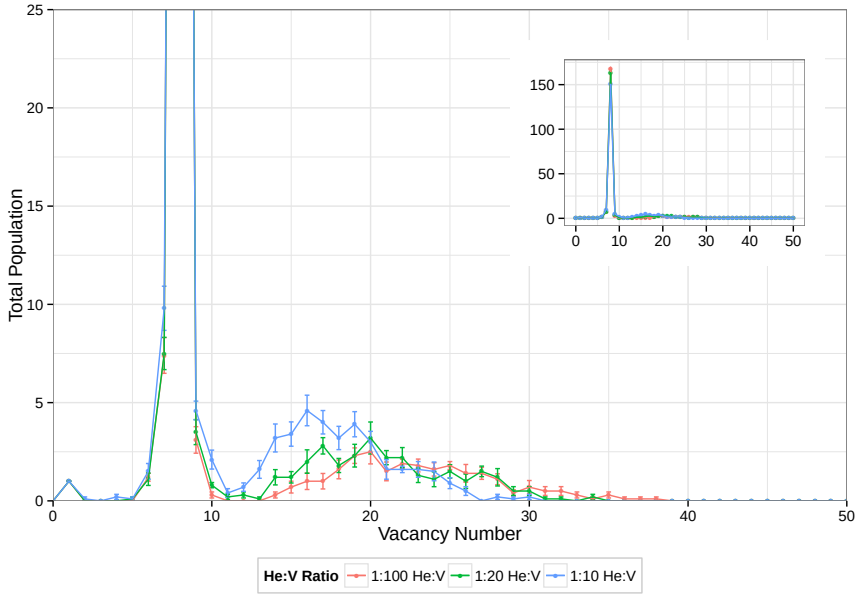



**Figure 4.21:** Final ($t \approx 90\,\mathrm{s}$) cluster size distribution for varying He:V ratios for variable dissociation energy model and limited cluster mobility. Clusters are aggregated based on number of vacancies.

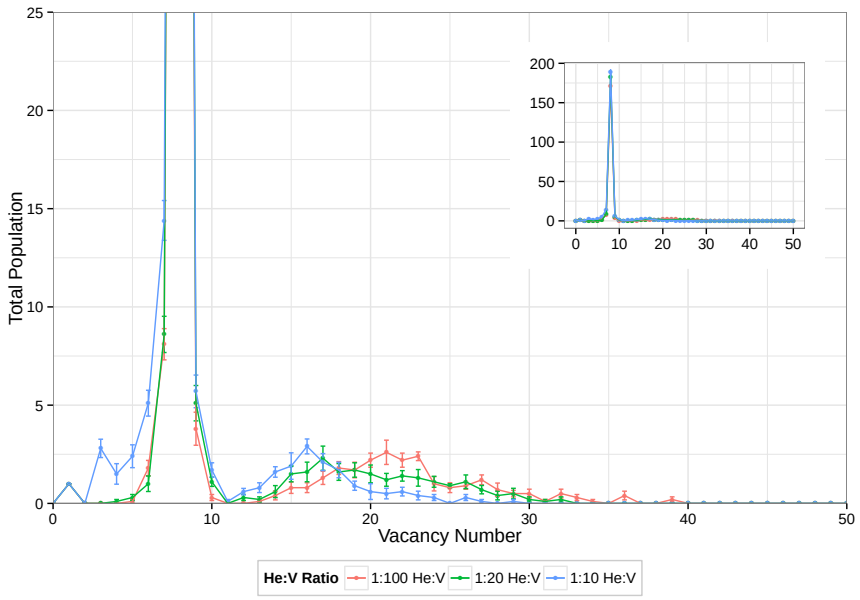
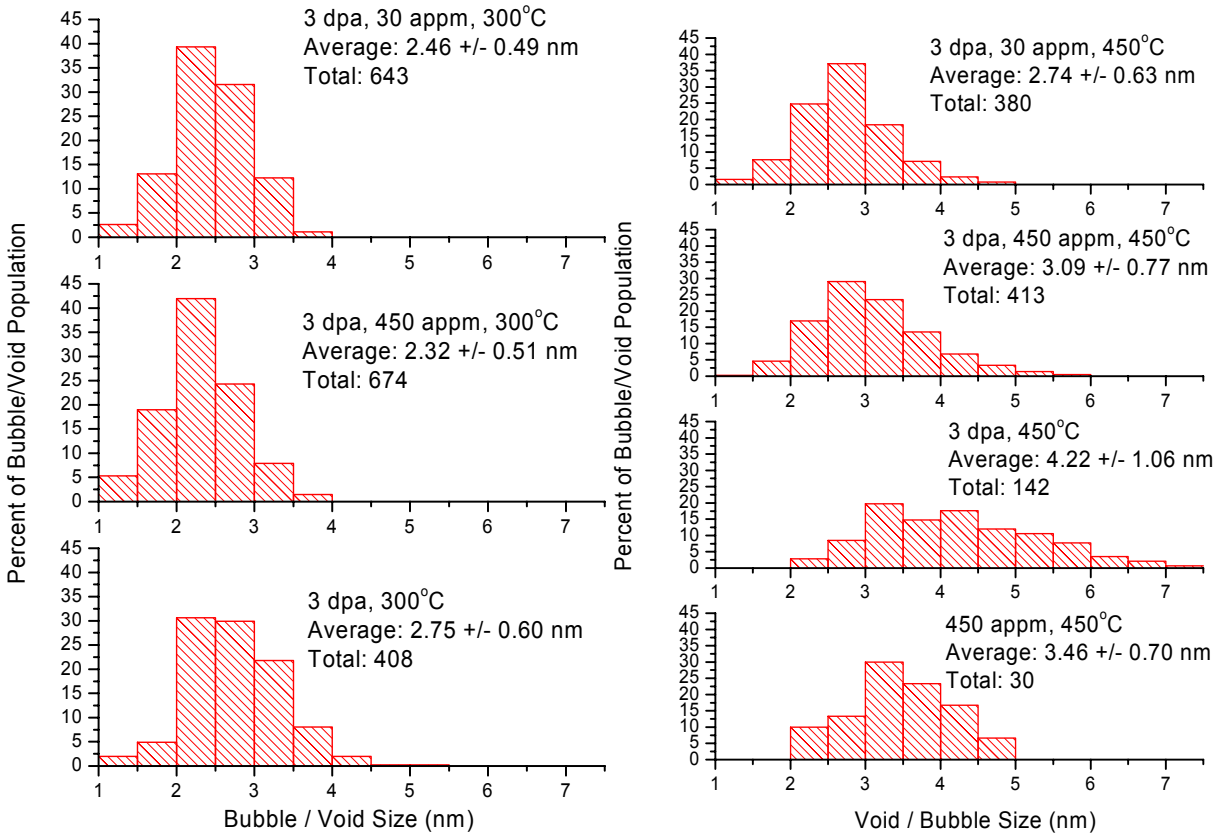



**Figure 4.22:** Final ($t \approx 90\,\mathrm{s}$) cluster size distribution for varying He:V ratios for variable dissociation energy model and noHe cluster mobility. Clusters are aggregated based on number of vacancies.

model (assumed for computational efficiency). This suggests that most clusters stayed small with a few growing up to and hitting the maximum size allowed in the system. As this behavior was only seen in the constant dissociation energy model results and not the variable dissociation energy model results, it is likely that the behavior is due to constant dissociation energy model in conjunction with the low helium concentration. In the constant dissociation energy model, the presence of helium strongly binds vacancies to their clusters, but vacancy-only clusters are only weakly bound, so in the case of low helium concentration, there are relatively few strongly bound clusters for the vacancies to bind to, and spend the bulk of their time forming and dissociating from smaller clusters. This was also seen computationally, as the lowest helium concentration configuration required several times as many simulation steps to reach the same point in real time. Given this, it was possible that the lowest concentration result would have resulted in a broad, low magnitude peak above $He_mV_{50}$ if the simulation system allowed them to form, resulting in confirmation of overall trend seen in the variable radius model and the experimental results. The simulations were rerun using a larger maximum cluster size, resulting in the more consistent distributions seen in figs. 4.24, 4.25 and 4.26.

From these trends it is concluded that the variable energy model produced distributions that were more consistent with the high size, low density distributions seen experimentally. The shape of the curves were significantly distorted by the peak at $V_8$, but if a similar energy set without the heightened $V_8$ dissociation energy, or if the system was allowed to anneal for longer




**(a)** 300 °C **(b)** 450 °C

**Figure 4.23:** Void size spectra for 3 dpa iron specimens irradiated at (a) 300 °C (b) 450 °C. The average void size and standard deviation is shown. The total number of voids counted is also given.[11]

times, the curve would likely redistribute into a more typical shape.

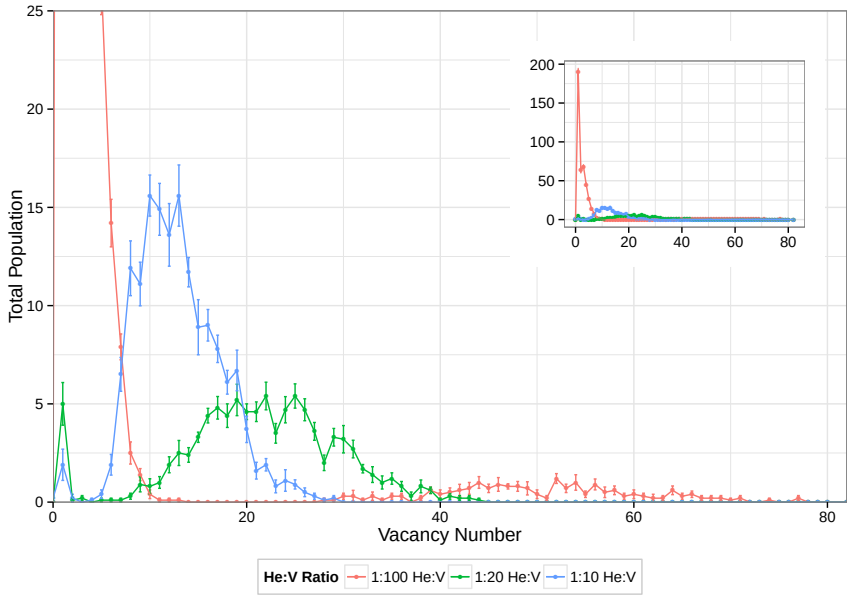



**Figure 4.24:** Final ($t \approx 1 \times 10^{-3}$ s) cluster size distribution for varying He:V ratios for constant dissociation energy model and full cluster mobility. Clusters are aggregated based on number of vacancies.

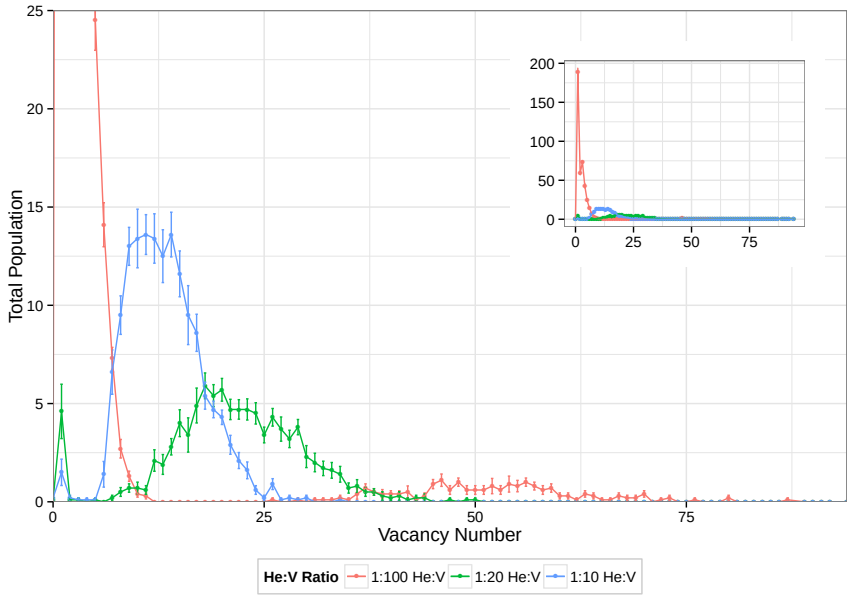



**Figure 4.25:** Final ($t \approx 1 \times 10^{-3}$ s) cluster size distribution for varying He:V ratios for constant dissociation energy model and limited cluster mobility. Clusters are aggregated based on number of vacancies.

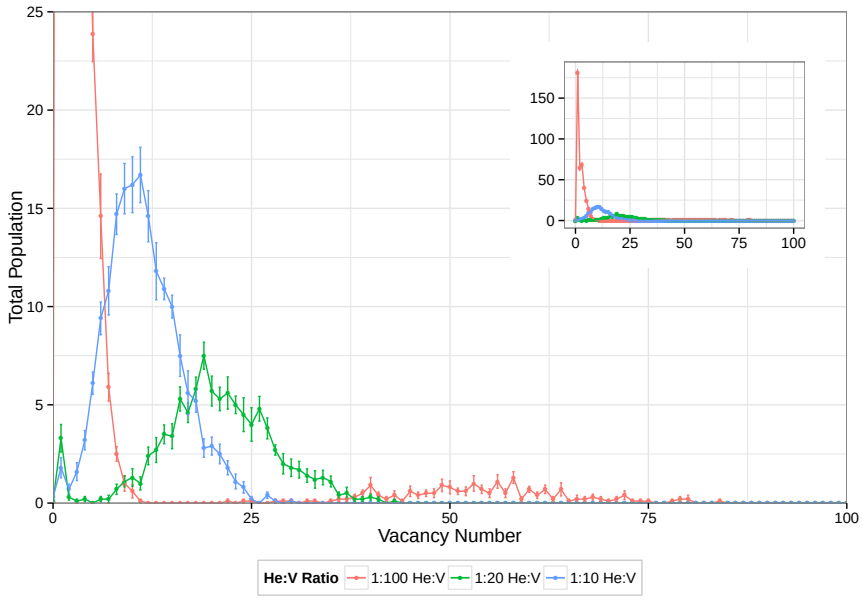



**Figure 4.26:** Final ($t \approx 1 \times 10^{-3}$ s) cluster size distribution for varying He:V ratios for constant dissociation energy model and noHe cluster mobility. Clusters are aggregated based on number of vacancies.

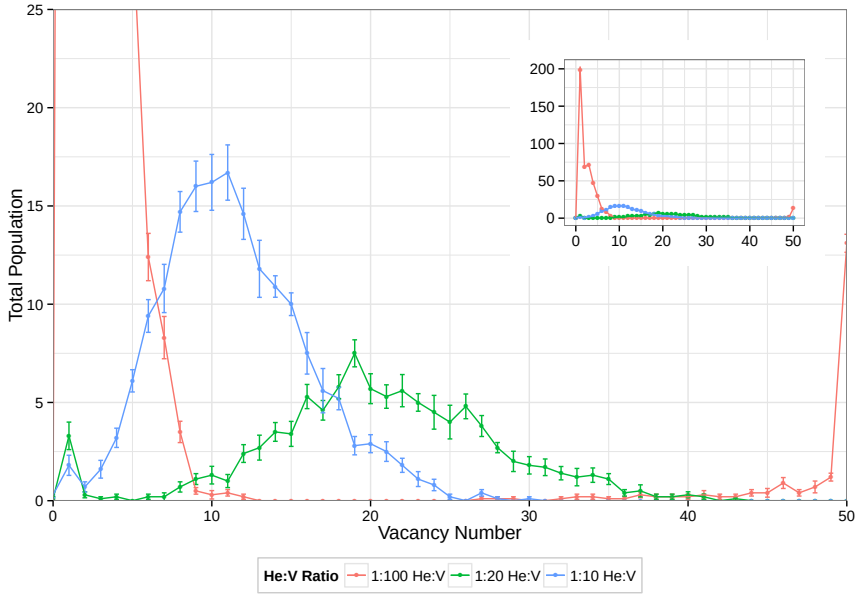



**Figure 4.27:** Final ($t \approx 1 \times 10^{-3}$ s) cluster size distribution for varying He:V ratios for constant dissociation energy model and noHe cluster mobility. Clusters are aggregated based on number of vacancies. Maximum cluster size set at $He_mV_{50}$, resulting in population peak as clusters saturated at the maximum.

## 4.6 Damage Results and Discussion

Most of the simulation conditions for the damage simulations are the same as those for the annealing simulations. The general properties and parameters for vacancies, interstitials, and helium are as described in section 4.1. The simulation system was a 125 × 125 × 125 BCC iron unit cell system with periodic boundary conditions. The temperature was set to 300 °C (~0.3$T_m$), and initial populations were randomly distributed throughout the system. 2000 vacancies (~0.05 at%) were introduced in all simulations. Due to simulation time constraints, helium was introduced at only the endpoints of the concentrations used in the annealing simulations, 1 % and 10 % of the vacancy concentration (also written as He:V ratios of 1:100, and 1:10, respectively) in order to study their effect on the various models. These simulations were again run for 12 × $10^6$ KMC steps, and were extended in most cases (typically in increments of 4 × $10^6$ steps) to allow for proper comparison at similar DPA levels. Each configuration was run with 10 different random initial configurations to obtain statistics. Distribution results are typically reported as averages with accompanying standard error bars.

### 4.6.1 Parameter Models

The model parameters used in the damage evolution simulations are a subset of those described earlier in sections 4.2 to 4.4, narrowed based on the annealing parameter exploration in section 4.5. From the radius model comparison, it was determined that the addition of helium dependence in the variable radius models did not significantly affect the distribution when compared with the non-helium dependent model, even with the highest concentration of helium present. Thus, only the "R(V)" model described in eq. (4.6) was considered. From the overall helium concentration comparison in section 4.5.6, the variable energy model described in section 4.3.2 produced distributions that were more consistent with the high size, low density distributions seen experimentally. Finally, from the cluster mobility comparison in section 4.5.5, it was determined that all three models should be compared under damage conditions.

### 4.6.2 Damage Parameters

To introduce damage into the system, Frenkel pairs are added to correspond to radiation damage events. The total number of Frenkel pairs added per event was given by the Norgett-Robinson-Torrens [75] formula:

$$\text{Displacements/event} = \frac{0.8E_D}{2E_d} \tag{4.11}$$

where $E_D$ is the energy of the primary knock-on atom (PKA) and $E_d$ is the displacement energy (40 eV for iron [76]). Using a PKA energy of 4 keV results in a damage rate of 40 FP/event. The damage event rate was taken to be $1 \times 10^2$ Hz, which corresponds to a damage rate of approximately $1 \times 10^{-3}$ dpa/s. The damage evolution was compared at multiple damage levels up to a total damage level of approximately 0.002 dpa (≈8000 FP).

### 4.6.3 Damage Level Comparison

Before comparing the effects of the different simulation parameters, the dpa evolution of the system is first explored. The 1:10 He:V system using the full mobility set and variable dissociation energy model is used as an example, with several DPA levels compared in fig. 4.28.

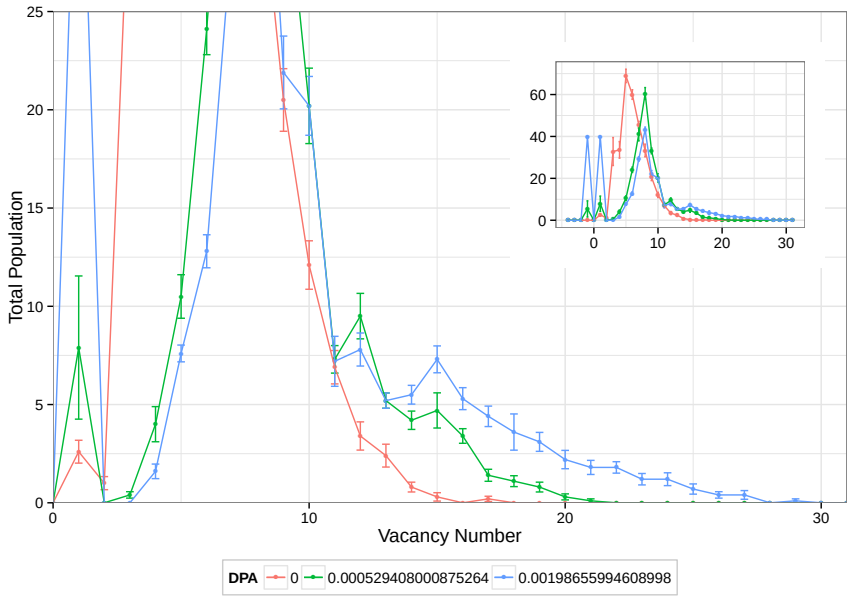



**Figure 4.28:** DPA evolution of damage system. DPA levels correspond roughly to times of $1.3 \times 10^{-3}$ s, 0.51 s and 1.9 s, respectively.

As was seen in the annealing simulations based on the same parameters (figs. 4.15 and 4.16), the size distribution shifts towards larger clusters as time/dpa evolves forward. These results also follow a similar form as seen in the annealing simulation in the same time frame (fig. 4.15), however there are several significant differences. First is the presence of a peak at $V_1$ and $-V_1$ ($I_1$). These peaks are caused by the introduction of multiple Frenkel pairs during each damage

event, and are somewhat artificial, as their size depends on when the simulation was stopped relative to the previous damage event. In the final DPA curve, the simulation ended relatively soon after the previous damage event, resulting in peaks corresponding the number of Frenkel pairs introduced at the damage event, while the earlier two curves were taken after the defects had more time diffuse and cluster. Second, and more importantly, the peaks in the curve have broadened and overlapped. Whereas in annealing case there was a sharp peak at $V_8$, dropping sharply at $V_{11}$, then peaking again at $V_{14}$ and falling off at $V_{20}$, in the final damage case the peak at $V_8$ is significantly lower and broader, and continues to drop as a single skewed peak down to $V_{30}$.

The behavior of the interstitials introduced during the damage production simulations is also considered here. The interstitial cluster populations can be seen in the full zoom plot overlays as negative size vacancy clusters. It is interesting to note that during these simulations, while the interstitials did cluster and interact with vacancies, they never grew to the large sizes seen in the $He_mV_n$ clusters. In the example shown in fig. 4.28, only interstitial clusters up to $I_4$ were present during information dump steps. In addition, there never appeared to be significant accumulation of interstitials, implying that most of the interstitials introduced during damage events eventually recombined with vacancies. However, even without accumulating, it is clear that their presence had a significant impact on vacancy cluster size distribution.

### 4.6.4 Cluster Mobility Comparison

In this section, the three cluster mobility models described in section 4.4 are compared with accumulating damage. As was done earlier for the annealing results, the cluster mobility models are compared first for a specific helium concentration in this section, then between helium concentrations in section 4.6.5.

Figures 4.29 to 4.31 show the initial ($t \approx 1.85 \times 10^{-3}$ s, DPA $\approx 0$) midpoint ($t \approx 0.51$ s, DPA $\approx 5.24 \times 10^{-4}$) and final ($t \approx 2$ s, DPA $\approx 2 \times 10^{-3}$) cluster size distributions for the 1:10 He:V ratio and the variable dissociation energy model. As in the annealing case, the three cluster mobility models overlapped almost completely at the initial point (before damage is introduced), but resulted in varying final distributions. However, as was discussed in the previous section, the introduction of damage served to broaden and overlap each of the mobility curves in the annealing case. Also as with the annealing results, the full and limited cluster mobility models mostly overlapped, while the immobile He-V cluster model resulted in lower density of larger clusters by comparison.

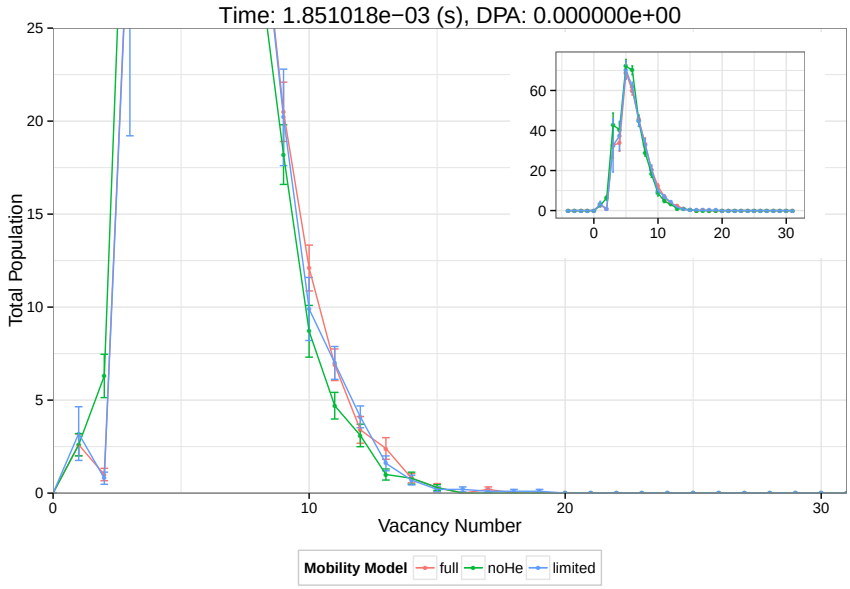



**Figure 4.29:** Initial ($t \approx 1.85 \times 10^{-3}$ s, DPA $\approx 0$) cluster size distribution for varying mobility models for 1:10 He:V ratio and variable dissociation energy model with introduced damage. Clusters are aggregated based on number of vacancies.

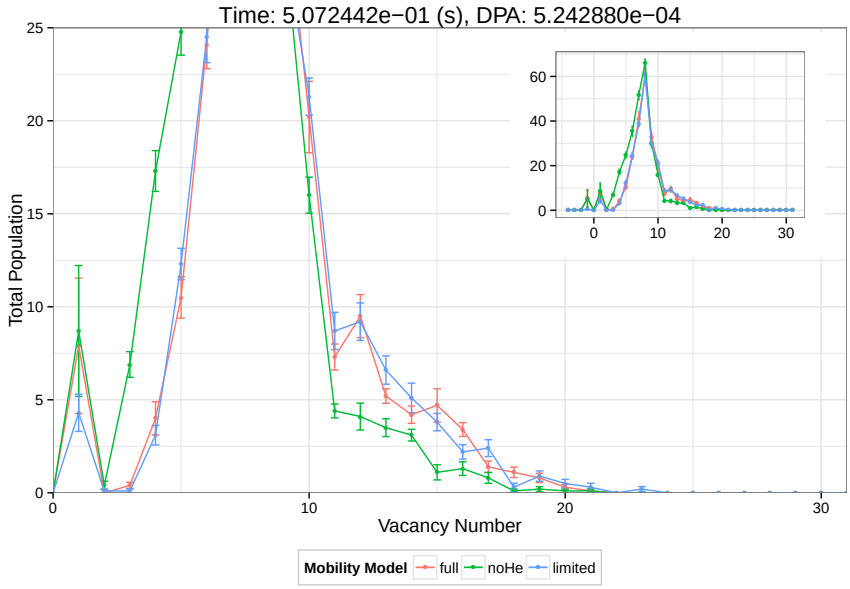



**Figure 4.30:** Midpoint ($t \approx 0.51$ s, DPA $\approx 5.24 \times 10^{-4}$) cluster size distribution for varying mobility models for 1:10 He:V ratio and variable dissociation energy model with introduced damage. Clusters are aggregated based on number of vacancies.

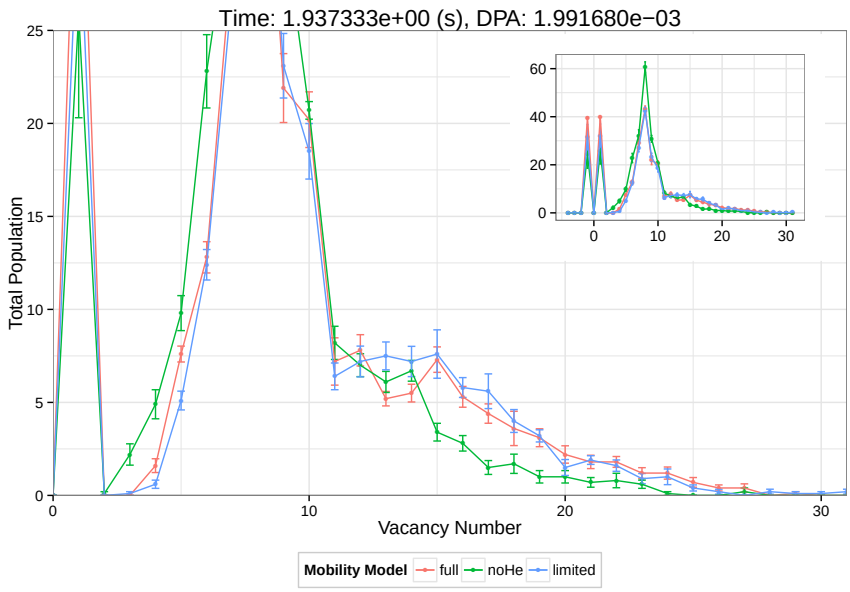



**Figure 4.31:** Final ($t \approx 2\,\mathrm{s}$, DPA $\approx 2 \times 10^{-3}$) cluster size distribution for varying mobility models for 1:10 He:V ratio and variable dissociation energy model with introduced damage. Clusters are aggregated based on number of vacancies.

### 4.6.5 Helium Concentration Comparison

Finally, the effect of helium concentration is compared for the variable migration energy model and cluster mobility model combinations. Since the damage simulations take considerably longer to perform than the annealing simulations, only the distributions of the two endpoint helium concentrations (He:V ratios of 1:100 and 1:10) are compared for each set of parameters. As was seen in earlier sections, while the different parameters resulted in significantly different simulation times, the results can be compared at similar damage levels.

Figures 4.32, 4.33 and 4.34 show the final distributions for the full mobility model, limited mobility model, and noHe mobility model, respectively. The variable dissociation energy models all followed the same general form seen in fig. 4.31. Consistent with the mobility comparison for the variable migration energy model, the full mobility model and limited mobility model result in similar trends. The strongest shift was observed in the noHe mobility model. Since $He_mV_n$ clusters are assumed to be immobile in this model, the increased helium concentration combined with the strong mobility trapping creates more small trapped clusters.

When comparing distributions of differing He:V ratios, the distributions all followed the same general trend: increasing the helium concentration shifts the size distribution down from smaller numbers of larger clusters to larger numbers of small clusters. This is likely due to the trapping effect of the helium. Vacancies bind to the helium which makes it more difficult

to dissociate or migration to find larger clusters. Rather than coalescing, these defect act as traps for mobile defects, resulting in a larger number of smaller clusters. This is consistent with the trend seen experimentally by Okuniewski [11] in fig. 4.23. These results are based on much higher defect concentrations (after 3 dpa) than used in these damage runs, however qualitatively the shift towards a higher density of smaller clusters is consistent.

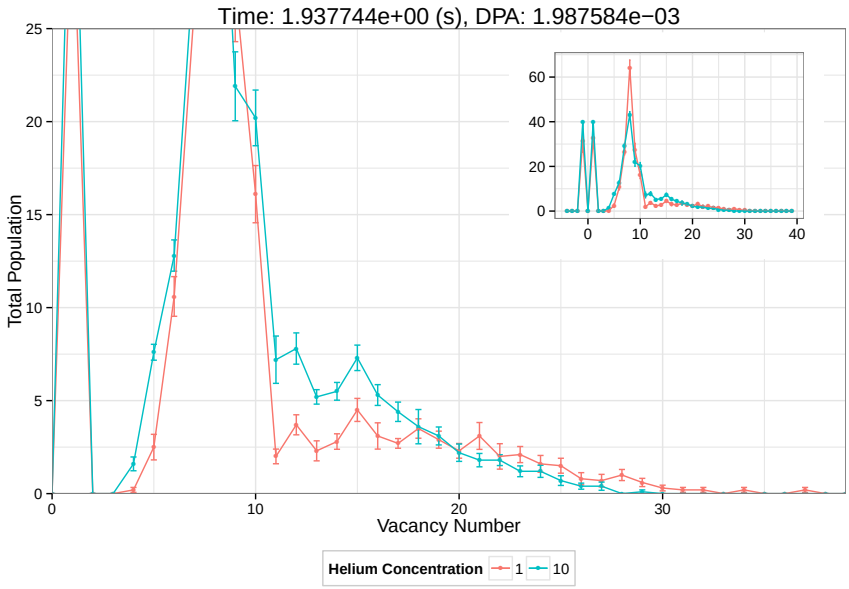



**Figure 4.32:** Final ($DPA \approx 2 \times 10^{-3}$) cluster size distribution for varying He:V ratios for variable dissociation energy model and full cluster mobility. Clusters are aggregated based on number of vacancies.

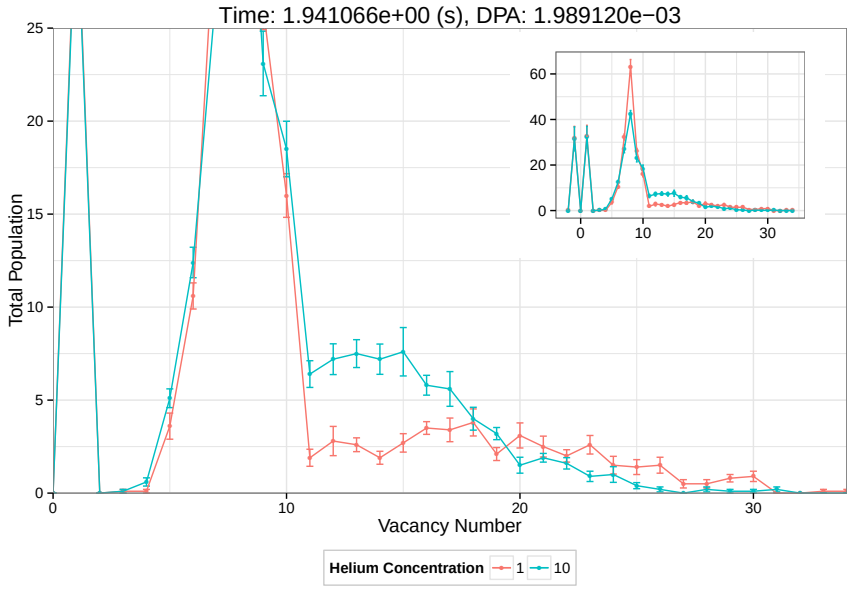



**Figure 4.33:** Final ($DPA \approx 2 \times 10^{-3}$) cluster size distribution for varying He:V ratios for variable dissociation energy model and limited cluster mobility. Clusters are aggregated based on number of vacancies.

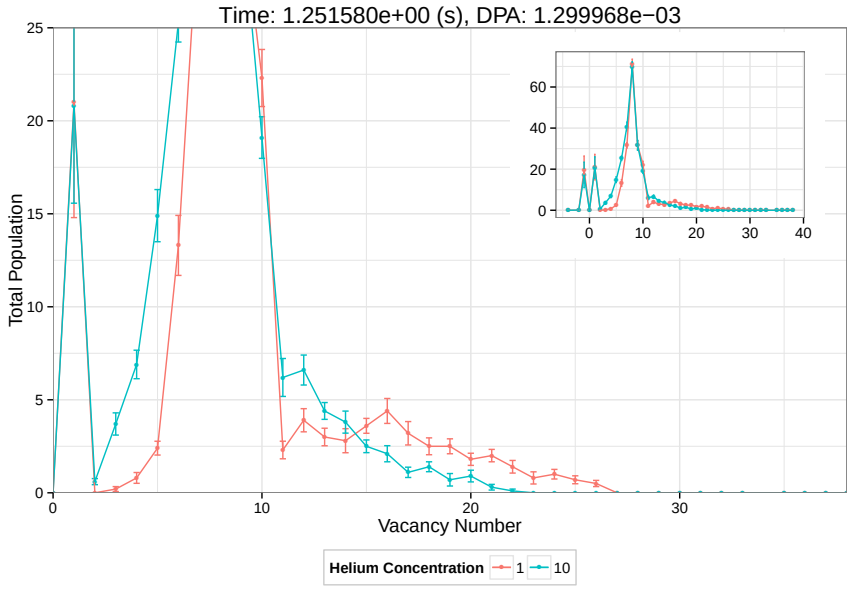



**Figure 4.34:** Final ($DPA \approx 1.3 \times 10^{-3}$) cluster size distribution for varying He:V ratios for variable dissociation energy model and noHe cluster mobility. Clusters are aggregated based on number of vacancies.

# Chapter 5

# Conclusions and Future Work

This work was designed with two primary goals. First, to develop a flexible KMC code capable of simulating the desired models, and second, to explore the modeling assumptions made in the previous KMC simulations in an attempt to determine their effects on cluster size distribution. The highlights of this work are presented in this section.

## 5.1 KMC Code Development

- A general form KMC code was designed and developed which included the following features:
  - Written in standard C++ using either standard or readily available libraries.
  - System configuration read in from input files.
  - End conditions based on KMC steps, simulation time, or DPA level.
  - XML data transport for straightforward programmatic generation of input files and parsing or output files.
  - Timeseries output to CSV, spatial information output in LAMMPS DUMP format.
  - Snapshot/resume functionality to extend previously run simulations or simulations that were cut short.
- A clustering model was implemented with the following functionality:
  - Frenkel pair recombination
  - Cluster formation
  - Cluster growth
  - Cluster shrinkage
  - Cluster coalescence

  - Cluster migration (assumed via surface diffusion)
  - One-dimensional migration
  - Cluster/particle dissociation
  - Defect production (including Frenkel pairs)

- Efficiency improvements were introduced and the code was benchmarked for performance with the following results:
  - Action set mapping reduced event catalog construction from linear time with respect to number of permitted entity types to nearly constant (constant time if no dissociation actions are present).
  - OpenMP parallelization implemented during event catalog construction resulted in linear improvement in runtime with respect to number of cores used. Diminishing returns found as number of processors increased.
  - Population tracking eliminated the need for iterating through the entity list during data dump steps.

- State-saving functionality was added to restarting simulations for a previously evolved state.

## 5.2 KMC Model Exploration

- Vacancy-only radius comparison
  - The larger constant radius model increases the cluster formation rate for small clusters ($v_2$,$v_3$), but does not help the growth of larger clusters.
  - Between the constant and variable interaction radius models, the variable radius model increases the initial formation and growth rate of larger clusters and results in a higher concentration of larger clusters compared to the constant radius model.
  - The R(V) better reproduces the cluster distribution trends found experimentally.

- Radius model comparison
  - In the constant radius model, particles spend a large number of KMC steps migrating around each other without interacting, resulting in significantly longer run times.

- The variable radius models produce size distributions which are more consistent with the large size low density distributions found experimentally.
- The constant radius model lagged behind the variable radius models in each case, and still resulted in more small clusters and fewer large clusters.

- Dissociation model comparison
  - The sharp peak at $V_8$ produced in the variable-energy model is likely due to the relatively high dissociation energy for $V_8$ clusters compared to surrounding cluster sizes.
  - The large jump in time scale produced in the constant-energy model is likely due to the high vacancy dissociation energy from $He_mV_n$ clusters relative to $V_n$ clusters.
  - Variable-energy model resulted in the desired shift towards a lower density of larger clusters when compared to the constant-energy model.

- Cluster mobility model comparison
  - With the variable migration energy model, the full and limited cluster mobility models mostly overlapped, while the immobile $He_mV_n$ cluster model resulted in lower density of smaller clusters by comparison.
  - With the constant migration energy model, the full and limited cluster mobility models mostly overlapped, while the immobile $He_mV_n$ cluster model resulted in lower density of smaller clusters by comparison.
  - The significant difference between the full and limited models that could appear at this stage is ignoring the effect of helium on the $He_mV_n$ clusters, however this change resulted in no significant change in the size distribution.
  - Since the presence of helium acts to slow down cluster migration as helium is added, the previous conclusion also shows that the helium content of the clusters remained relatively low.

- Helium concentration comparison
  - Consistent with the mobility comparison for the variable migration energy model, the full mobility model and limited mobility model result in similar trends.
  - The strongest shift was observed in the noHe mobility model. Since $He_mV_n$ clusters are assumed to be immobile in this model, the increased helium concentration combined with the strong mobility trapping creates more small trapped clusters.

  - When comparing distributions of differing He:V ratios, the distributions all followed the same general trend: increasing the helium concentration shifts the size distribution down from smaller numbers of larger clusters to larger numbers of small clusters, likely due to the trapping effect of the helium.
  - Results are qualitatively consistent with the trend seen experimentally by Okuniewski [11].
  - Unlike in the variable migration energy results, the distinction between the full mobility model and limited mobility model results compared with the noHe mobility model was much less significant for larger clusters in the constant migration energy results.
  - While the 1:10 and 1:20 He:V ratio distributions formed broad peaks with a small speak at the lower end corresponding to mobile defects, the 1:100 He:V ratio distribution formed a sharp peak at the lower end with the population consisting primarily of small clusters.
  - In the constant dissociation energy model, the presence of helium strongly binds vacancies to their clusters, but vacancy-only clusters are only weakly bound, so in the case of low helium concentration, there are relatively few strongly bound clusters for the vacancies to bind to, and spend the bulk of their time forming and dissociating from smaller clusters.
  - Overall, the variable energy model produced distributions that were more consistent with the high size, low density distributions seen experimentally.

- Damage level comparison
  - In the final DPA curve, the simulation ended relatively soon after the previous damage event, resulting in somewhat artificial peaks corresponding the number of Frenkel pairs introduced at the damage event, while the earlier two curves were taken after the defects had more time diffuse and cluster.
  - The presence of damage causes the peaks observed in the annealing results to broaden and overlap.
  - Interstitials did not form large clusters, in most cases no larger than $I_4$.
  - There was no significant accumulation of interstitials or additional vacancies.
  - Even without SIA defect accumulation, the presence of damage had significant impact on vacancy cluster size distribution.

- Damage mobility model comparison
  - The introduction of damage served to broaden and overlap each of the mobility curves in the annealing case.
  - The full and limited cluster mobility models mostly overlapped, while the immobile He-V cluster model resulted in lower density of larger clusters by comparison.
- Damage helium concentration comparison
  - Consistent with the mobility comparison for the variable migration energy model, the full mobility model and limited mobility model result in similar trends.
  - The strongest shift was observed in the noHe mobility model. Since $He_mV_n$ clusters are assumed to be immobile in this model, the increased helium concentration combined with the strong mobility trapping creates more small trapped clusters.
  - When comparing distributions of differing He:V ratios, the distributions all followed the same general trend: increasing the helium concentration shifts the size distribution down from smaller numbers of larger clusters to larger numbers of small clusters, likely due to the trapping effect of the helium.
  - Results are qualitatively consistent with the trend seen experimentally by Okuniewski [11].

## 5.3 Future Work

If this work is continued in the future, it would be interesting to develop the following avenues:

- Integrate OpenMP 4.0 functionality to avoid list-to-vector copying and improve event catalog construction.
- Develop entity-event mapping, similar to action-entity mapping, to allow for in-place updates of the event catalog, rather than complete reconstruction.
- Explore simulations using variable energy dissociation model that does not contain a heightened dissociation energy for one cluster relative to its neighbors.
- Explore in damage-anneal sequences to better see their combined effects.

# References


[1] L. R. GREENWOOD, B. M. OLIVER, S. OHNUKI, K. SHIBA, Y. KOHNO, A. KOHYAMA, J. P. ROBERTSON, J. W. MEADOWS, and D. S. GELLES. Accelerated helium and hydrogen production in 54Fe doped alloys - measurements and calculations for the FIST experiment. *Journal of Nuclear Materials*, **283-287**: 1438–1442, 2000.

[2] F. W. CLINARD. First wall materials problems in fusion reactors. *Journal of Vacuum Science and Technology*, **12**: 510, 1975.

[3] J. ROTHAUT, H. SCHROEDER, and H. ULLMAIER. The growth of helium bubbles in stainless steel at high temperatures. *Philosophical Magazine A: Physics of Condensed Matter, Structure, Defects and Mechanical Properties*, **47**: 781–795, 1983.

[4] S. JITSUKAWA, A. KIMURA, A. KOHYAMA, R. KLUEH, A. TAVASSOLI, B. VAN DER SCHAAF, G. ODETTE, J. RENSMAN, M. VICTORIA, and C. PETERSEN. Recent results of the reduced activation ferritic/martensitic steel development. *Journal of Nuclear Materials*, **329-333, Part A**: 39–46, 2004.

[5] L. MANSUR and M. GROSSBECK. Mechanical property changes induced in structural alloys by neutron irradiations with different helium to displacement ratios. *Journal of Nuclear Materials*, **155-157, Part 1**: 130–147, 1988.

[6] H. ULLMAIER. Introductory remarks - helium in metals. *Radiation Effects*, **78**: 1–10, 1983.

[7] H. TRINKAUS and B. N. SINGH. Helium accumulation in metals during irradiation - where do we stand? *Journal of Nuclear Materials*, **323**: 229–242, 2003.

[8] M. GROSSBECK and K. LIU. High-temperature fatigue life of type 316 stainless steel containing irradiation induced helium. *Journal of Nuclear Materials*, **104**: 853–857, 1981.

[9] M. GROSSBECK and J. HORAK. Irradiation creep in type 316 stainless steel and US PCA with fusion reactor He/dpa levels. *Journal of Nuclear Materials*, **155**: 1001–1005, 1988.

[10] H. SCHROEDER and P. BATFALSKY. The dependence of the high temperature mechanical properties of austenitic stainless steels on implanted helium. *Journal of Nuclear Materials*, **117**: 287–294, 1983.

[11] M. OKUNIEWSKI. *Irradiation-Induced Microstructural Evolution and Mechanical Properties in Iron With and Without Helium*. PhD Thesis. University of Illinois, Urbana-Champaign, 2008

[12] R. SINGH. "Chapter 4 - Structure of Materials" in: *Applied Welding Engineering*. Boston: Butterworth-Heinemann, 2012. 23–32 ISBN: 978-0-12-391916-8

[13] D. E. RIMMER and A. H. COTTRELL. The solution of inert gas atoms in metals. *Philosophical Magazine*, **2**: 1345–1353, 1957.

[14] C.-C. FU and F. WILLAIME. Ab initio study of helium in alpha -Fe : Dissolution, migration, and clustering with vacancies. *Physical Review B*, **72**: 064117, 2005.

[15] T. SELETSKAIA, Y. OSETSKY, R. E. STOLLER, and G. M. STOCKS. Magnetic Interactions Influence the Properties of Helium Defects in Iron. *Physical Review Letters*, **94**: 046403, 2005.

[16] X. T. ZU, L. YANG, F. GAO, S. M. PENG, H. L. HEINISCH, X. G. LONG, and R. J. KURTZ. Properties of helium defects in bcc and fcc metals investigated with density functional theory. *Physical Review B*, **80**: 054104, 2009.

[17] M. SAMARAS. Multiscale Modelling: the role of helium in iron. *Materials Today*, **12**: 46–53, 2009.

[18] W. D. WILSON, C. L. BISSON, and M. I. BASKES. Self-trapping of helium in metals. *Physical Review B*, **24**: 5616, 1981.

[19] T. SELETSKAIA, Y. OSETSKIY, R. STOLLER, and G. STOCKS. Development of a Fe-He interatomic potential based on electronic structure calculations. *Journal of Nuclear Materials*, **367–370, Part A**: 355–360, 2007.

[20] N. JUSLIN and K. NORDLUND. Pair potential for Fe-He. *Journal of Nuclear Materials*, **382**: 143–146, 2008.

[21] G. J. ACKLAND, D. J. BACON, A. F. CALDER, and T. HARRY. Computer simulation of point defect properties in dilute Fe-Cu alloy using a many-body interatomic potential. *Philosophical Magazine A*, **75**: 713–732, 1997.

[22] S. L. DUDAREV and P. M. DERLET. A 'magnetic' interatomic potential for molecular dynamics simulations. *Journal of Physics: Condensed Matter*, **17**: 7097–7118, 2005.

[23] M. I. MENDELEV, S. HAN, D. J. SROLOVITZ, G. J. ACKLAND, D. Y. SUN, and M. ASTA. Development of new interatomic potentials appropriate for crystalline and liquid iron. *Philosophical Magazine*, **83**: 3977–3994, 2003.

[24] D. E. BECK. A new interatomic potential function for helium. *Molecular Physics: An International Journal at the Interface Between Chemistry and Physics*, **14**: 311–315, 1968.

[25] D. NGUYEN-MANH, A. P. HORSFIELD, and S. L. DUDAREV. Self-interstitial atom defects in bcc transition metals: Group-specific trends. *Physical Review B*, **73**: 020101, 2006.

[26] C. DOMAIN and C. S. BECQUART. Ab initio calculations of defects in Fe and dilute Fe-Cu alloys. *Physical Review B*, **65**: 024103, 2001.

[27] K. MORISHITA, R. SUGANO, B. D. WIRTH, and T. DIAZ DE LA RUBIA. Thermal stability of helium-vacancy clusters in iron. *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, **202**: 76–81, 2003.

[28] L. VENTELON, B. WIRTH, and C. DOMAIN. Helium-self-interstitial atom interaction in alpha-iron. *Journal of Nuclear Materials*, **351**: 119–132, 2006.

[29] L. YANG, X. T. ZU, F. GAO, H. L. HEINISCH, and R. J. KURTZ. Effects of Fe-He potential on primary damage formation in Fe-1% He. *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, **267**: 3046–3049, 2009.

[30] B. WIRTH, G. ODETTE, J. MARIAN, L. VENTELON, J. YOUNG-VANDERSALL, and L. ZEPEDA-RUIZ. Multiscale modeling of radiation damage in Fe-based alloys in the fusion environment. *Journal of Nuclear Materials*, **329–333, Part A**: 103–111, 2004.

[31] K. MORISHITA, R. SUGANO, I. IWAKIRI, N. YOSHIDA, and A. KIMURA "Thermal Helium Desorption from alpha-Iron" Fourth Pacific Rim International Conference on Advanced Materials and Processing Honolulu, Hawaii, Dec. 2001

[32] C.-C. FU and F. WILLAIME. Interaction between helium and self-defects in [alpha]-iron from first principles. *Journal of Nuclear Materials*, **367-370**: 244–250, 2007.

[33] G. LUCAS and R. SCHÄUBLIN. Stability of helium bubbles in alpha-iron: A molecular dynamics study. *Journal of Nuclear Materials*, **386-388**: 360–362, 2009.

[34] H. TRINKAUS. Energetics and formation kinetics of helium bubbles in metals. *Radiation Effects*, **78**: 189–211, 1983.

[35] C. A. WALSH, J. YUAN, and L. M. BROWN. A procedure for measuring the helium density and pressure in nanometre-sized bubbles in irradiated materials using electron-energy-loss spectroscopy. *Philosophical Magazine A*, **80**: 1507–1543, 2000.

[36] Y. OSETSKY and D. BACON. Comparison of void strengthening in fcc and bcc metals: Large-scale atomic-level modelling. *Materials Science and Engineering: A*, **400–401**: 374–377, 2005.

[37] Y. OSETSKY and D. BACON. Void and precipitate strengthening in $\alpha$-iron: what can we learn from atomic-level modelling? *Journal of Nuclear Materials*, **323**: 268–280, 2003.

[38] S. M. HAFEZ HAGHIGHAT, G. LUCAS, and R. SCHÄUBLIN. State of a pressurized helium bubble in iron. *EPL (Europhysics Letters)*, **85**: 60008, 2009.

[39] L. YANG, X. T. ZU, H. Y. XIAO, F. GAO, K. Z. LIU, H. L. HEINISCH, R. J. KURTZ, and S. Z. YANG. Temperature effects on He bubbles production due to cascades in [alpha]-iron. *Materials Science and Engineering: A*, **427**: 343–347, 2006.

[40] L. YANG, X. T. ZU, H. Y. XIAO, F. GAO, H. L. HEINISCH, R. J. KURTZ, and K. Z. LIU. Atomistic simulation of helium-defect interaction in alpha-iron. *Applied Physics Letters*, **88**: 091915–3, 2006.

[41] L. YANG, X. T. ZU, H. Y. XIAO, F. GAO, H. L. HEINISCH, R. J. KURTZ, Z. G. WANG, and K. Z. LIU. Stability of helium clusters during displacement cascades. *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, **255**: 63–67, 2007.

[42] L. YANG, X. T. ZU, H. Y. XIAO, F. GAO, H. L. HEINISCH, and R. J. KURTZ. Defect production and formation of helium-vacancy clusters due to cascades in [alpha]-iron. *Physica B: Condensed Matter*, **391**: 179–185, 2007.

[43] R. SCHAUBLIN, J. HENRY, and Y. DAI. Helium and point defect accumulation: (i) microstructure and mechanical behaviour. *Comptes Rendus Physique*, **9**: 389–400, 2008.

[44] J. PU, L. YANG, X. T. ZU, and F. GAO. A molecular dynamics study of helium bubble stability during high-energy displacement cascades in [alpha]-iron. *Physica B: Condensed Matter*, **398**: 65–70, 2007.

[45] J. BIERSACK and J. ZIEGLER. Refined universal potentials in atomic collisions. *Nuclear Instruments and Methods in Physics Research*, **194**: 93–100, 1982.

[46] A. OAKS, D. YUN, B. YE, W.-Y. CHEN, and J. F. STUBBINS. Kinetic Monte Carlo model of defect transport and irradiation effects in La-doped CeO2. *Journal of Nuclear Materials*, **414**: 145–149, 2011.

[47] J. E. GENTLE. *Random number generation and Monte Carlo methods*. English New York [u.a.]: Springer, 2005. ISBN: 0387001786 9780387001784

[48] S. PLIMPTON *LAMMPS Dump Format* http://lammps.sandia.gov/doc/dump.html 2012

[49] S. PLIMPTON *Pizza.py Toolkit* http://pizza.sandia.gov/ 2012

[50] C. S. DEO, M. A. OKUNIEWSKI, S. G. SRIVILLIPUTHUR, S. A. MALOY, M. I. BASKES, M. R. JAMES, and J. F. STUBBINS. Helium bubble nucleation in bcc iron studied by kinetic Monte Carlo simulations. *Journal of Nuclear Materials*, **361**: 141–148, 2007.

[51] C. J. ORTIZ and M. J. CATURLA. Cascade damage evolution: rate theory versus kinetic Monte Carlo simulations. *Journal of Computer-Aided Materials Design*, **14**: 171–181, 2007.

[52] C. DOMAIN, C. BECQUART, and L. MALERBA. Simulation of radiation damage in Fe alloys: an object kinetic Monte Carlo approach. *Journal of Nuclear Materials*, **335**: 121–145, 2004.

[53] M. CATURLA, N. SONEDA, E. ALONSO, B. WIRTH, T. DÍAZ DE LA RUBIA, and J. PERLADO. Comparative study of radiation damage accumulation in Cu and Fe. *Journal of Nuclear Materials*, **276**: 13–21, 2000.

[54] H. TRINKAUS, B. SINGH, and S. GOLUBOV. Progress in modelling the microstructural evolution in metals under cascade damage conditions. *Journal of Nuclear Materials*, **283–287, Part 1**: 89–98, 2000.

[55] N. SONEDA, S. ISHINO, A. TAKAHASHI, and K. DOHI. Modeling the microstructural evolution in bcc-Fe during irradiation using kinetic Monte Carlo computer simulation. *Journal of Nuclear Materials*, **323**: 169–180, 2003.

[56] B. L. EYRE, R. C. PERRIN, and D. M. MAHER. Electron microscope image contrast from small dislocation loops. II. Application of theoretical predictions to defect analysis in irradiated metals. *Journal of Physics F: Metal Physics*, **7**: 1371, 1977.

[57] Z. ZHOU, M. L. JENKINS, S. L. DUDAREV, A. P. SUTTON, and M. A. KIRK. Simulations of weak-beam diffraction contrast images of dislocation loops by the many-beam Howie-Basinski equations. *Philosophical Magazine*, **86**: 4851–4881, 2006.

[58] A. VEHANEN, P. HAUTOJÄRVI, J. JOHANSSON, J. YLI-KAUPPILA, and P. MOSER. Vacancies and carbon impurities in $\alpha$- iron: Electron irradiation. *Physical Review B*, **25**: 762–780, 1982.

[59] C.-C. FU, J. D. TORRE, F. WILLAIME, J.-L. BOCQUET, and A. BARBU. Multiscale modelling of defect kinetics in irradiated iron. *Nature Materials*, **4**: 68–74, 2005.

[60] N. SONEDA and T. D. DE LA RUBIA. Defect production, annealing kinetics and damage evolution in $\alpha$-Fe: An atomic-scale computer simulation. *Philosophical Magazine A*, **78**: 995–1019, 1998.

[61] F. GAO, D. J. BACON, A. V. BARASHEV, and H. L. HEINISCH. Kinetic Monte Carlo Annealing Simulation of Damage Produced by Cascades in Alpha-Iron. *MRS Online Proceedings Library*, **540**: 1998.

[62] J. MARIAN, B. D. WIRTH, J. M. PERLADO, T. D. DE LA RUBIA, R. SCHÄUBLIN, D. LODI, M. HERNÁNDEZ, G. DE DIEGO, D. G. BRICEÑO, and R. E. STOLLER. Direct Comparison between Modeling and Experiment: an $\alpha$-Fe Ion Implantation Study. *MRS Online Proceedings Library*, **650**: R3.2.1–6, 2000.

[63] S. TAKAKI, J. FUSS, H. KUGLERS, U. DEDEK, and H. SCHULTZ. The resistivity recovery of high purity and carbon doped iron following low temperature electron irradiation. *Radiation Effects*, **79**: 87–122, 1983.

[64] H. MAETA, F. ONO, and T. KITTAKA. Point Defects and Induced Magnetic Anisotropy in Fast Neutron Irradiated Iron. *Journal of the Physical Society of Japan*, **53**: 4353–4358, 1984.

[65] Y. N. OSETSKY, D. J. BACON, A. SERRA, B. N. SINGH, and S. I. GOLUBOV. Stability and mobility of defect clusters and dislocation loops in metals. *Journal of Nuclear Materials*, **276**: 65–77, 2000.

[66] B. D. WIRTH, G. R. ODETTE, D. MAROUDAS, and G. E. LUCAS. Energetics of formation and migration of self-interstitials and self-interstitial clusters in $\alpha$-iron. *Journal of Nuclear Materials*, **244**: 185–194, 1997.

[67] Y. N. OSETSKY, M. VICTORIA, A. SERRA, S. I. GOLUBOV, and V. PRIEGO. Computer simulation of vacancy and interstitial clusters in bcc and fcc metals. *Journal of Nuclear Materials*, **251**: 34–48, 1997.

[68] F. GAO, G. HENKELMAN, W. J. WEBER, L. R. CORRALES, and H. JÓNSSON. Finding possible transition states of defects in silicon-carbide and alpha-iron using the dimer method. *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, **202**: 1–7, 2003.

[69] A. HARDOUIN DUPARC, C. MOINGEON, N. SMETNIANSKY-DE-GRANDE, and A. BARBU. Microstructure modelling of ferritic alloys under high flux 1 MeV electron irradiations. *Journal of Nuclear Materials*, **302**: 143–155, 2002.

[70] F. MAURY, M. BIGET, P. VAJDA, A. LUCASSON, and P. LUCASSON. Anisotropy of defect creation in electron-irradiated iron crystals. *Physical Review B*, **14**: 5303–5313, 1976.

[71] M. CATURLA and C. ORTIZ. Effect of self-interstitial cluster migration on helium diffusion in iron. *Journal of Nuclear Materials*, **362**: 141–145, 2007.

[72] D. BACON, F. GAO, and Y. OSETSKY. The primary damage state in fcc, bcc and hcp metals as seen in molecular dynamics simulations. *Journal of Nuclear Materials*, **276**: 1–12, 2000.

[73] V. BORODIN and P. VLADIMIROV. Diffusion coefficients and thermal stability of small helium-vacancy clusters in iron. *Journal of Nuclear Materials*, **362**: 161–166, 2007.

[74] C.-C. FU, F. WILLAIME, and P. ORDEJON. Stability and Mobility of Mono- and Di-Interstitials in alpha-Fe. *Physical Review Letters*, **92**: 175503, 2004.

[75] M. J. NORGETT, M. T. ROBINSON, and I. M. TORRENS. A proposed method of calculating displacement dose rates. *Nuclear Engineering and Design*, **33**: 50–54, 1975.

[76] R. E. STOLLER, G. R. ODETTE, and B. D. WIRTH. Primary damage formation in bcc iron. *Journal of Nuclear Materials*, **251**: 49–60, 1997.

[77] A. STUKOWSKI. Visualization and analysis of atomistic simulation data with OVITO – the Open Visualization Tool. *Modelling and Simulation in Materials Science and Engineering*, **18**: 015012, 2010.